Numerical Modeling of Multi-Fan Air Distribution Systems for Research Laboratories
(Draft Report)

NETSAL & ASSOCIATES
16182 Mount Lowe Circle
Fountain Valley, CA 92708

Principal Investigator:

September 25, 1998
Robert J. Tsal, Ph.D. **Date**

NUMERICAL MODELING OF MULTI-FAN AIR DISTRIBUTION SYSTEMS FOR RESEARCH LABORATORIES

EXECUTIVE SUMMARY **3**

ABSTRACT **6**

1. INTRODUCTION **6**

2. PROBLEM DEFINITION **8**

2.1 TOPOLOGY OF AIR FLOW SYSTEMS 8
2.2 HYDRAULICS OF AIR FLOW SYSTEMS 12
2.2.1 Ducts *12*
2.2.2 Fittings *16*
2.2.3 Infiltration and exfiltration *16*
2.2.4 Fans *16*
2.2.5 Fume hoods *18*
2.2.6 Air flow systems *18*

3. MATHEMATICAL AND ENGINEERING METHODS FOR NETWORK SIMULATION **23**

3.1 BRANCHED SYSTEMS 23
3.2 CYCLE SYSTEMS 28
3.2.2 Newton's method *29*
3.2.3 Broyden's method *33*
3.2.4 Merit functions: nonlinear equations as optimal problem *35*
3.2.5 Global affine invariant Newton's technique *35*
3.2.6 Tensor methods *43*
3.2.7 Methods based on homotopy *52*
3.2.8 Solving linear equations *61*
3.3 CONTROL SYSTEMS 72
3.4 FINAL DISCUSSION 78

4. PROBLEM FORMALIZATION **78**

4.1 TOPOLOGY 79
4.2 CONTROL FUNCTION 91
4.3 ELEMENTS 91

5. MAIN FLOW CHART **97**

6. PRINCIPLES OF SOFTWARE DEVELOPMENT **101**

7. SOFTWARE IMPLEMENTATION PLAN **102**

8. CONCLUSION **105**

9. REFERENCES AND BIBLIOGRAPHY **106**

EXECUTIVE SUMMARY

Studies show that air conditioning systems are one of the major energy consumers in laboratories. The cost of air conditioning systems can exceed 30% of the total first cost for the entire laboratory building. Proper system design is one of the efficient ways of reducing the life cycle cost and enhancing the performance of air conditioning systems. Depending on requirements, laboratory spaces must be either under positive or under negative pressure. If infiltration or exfiltration exceeds design limits it may result in room over- or under-pressurization, loss of pressure zoning, substantial heat and energy loss, considerable egress, dust and dirt carryover, and airflow destabilization. Actual operating conditions in a laboratory vary and can result in imbalance, and airflow destabilization beyond design limits. Such conditions can be inefficient, unsafe, and negatively impact the work being done in the laboratories.

Engineers have no practical technique to simulate actual airflow, pressures, fan operating points and the effect of system control in multi-fan laboratory air conditioning systems. That leads to unexpected conditions during operation of the system.

The purpose of this project is to define the problem and to develop the theory for numerical modeling of multi-fan systems serving multiple laboratories including determination of pressure in the workspaces and the position of control elements. This modeling will serve both variable air volume (VAV) and constant volume (CV) systems.

The necessity for simulation appears in many HVAC designs, such as determining operating performance, investigating system stability under different operating conditions, retrofitting, emergency conditions and accidents, fire/smoke protection, pressure and airflow balancing after system modification, and equipment failure.

Multiple laboratories located in the same building are usually served by one or more common supply systems and multiple exhaust systems, individual, manifolded, centralized or combined. Topologically the laboratory air distribution systems can be represented by a number of isomorphic tree-graphs connected at the terminals.

The general purpose of the project is to develop and implement a practical tool which is a computer code that can be used by HVAC engineers in studying, designing, and retrofitting of air distribution systems in research laboratories. Numerical modeling of multi-fan air distribution systems for research laboratories is the first part of this project.

The proposed computer program, named Bellair, will allow HVAC engineers to calculate actual airflow, pressures, and fan operating points in laboratory multi-fan air distribution systems as well as static pressure in laboratory spaces at different operating conditions where space pressurization, confinement zoning, and flow/pressure stability are the most important requirements. The computer simulation program will allow to play "what-if" scenarios which are most important to provide flexibility for laboratories with changing technology.

The computer program Bellair will be capable of modeling the control of each fan or damper from a

sensor located in any place of the system selected by the user including ducts, laboratories, cabinets, exhaust stack discharge, etc. The control parameters could be static pressure, total pressure, airflow volume, air velocity, temperature, and pressure or temperature difference.

Network simulation problem is considered to be solved when the iteration process that describes steady-state hydraulic and control conditions is converged. Such problem requires a solution for a large system of simultaneous nonlinear algebraic equations. Therefore one of the main goals of this project is the selection of a numerical method for solving such equations. Numerous methods capable of solving nonlinear algebraic equations for both branched and cycled networks simulation are studied. This includes for branched systems: Equivalent Nozzles, Unit Flow, Duct Characteristic, Equivalent Resistance, Steepest Descent, and T-Method with air leakage. As for cycle systems the more promising approaches are the family of Newton Affine Invariant methods, the Tensor methods, and the Homotopy methods. If a good initial guess is known, the best choice is one of the Newton Affine Invariant methods. For a comparatively large problem, when time for solution is significant, the Tensor methods are a good choice. For the most practical cases, when either: (1) time is deficient to search for a good initial guess, (2) automation is needed for different types of technical solutions evaluation, or (3) parametric evaluation is necessary, Homotopy methods are the best. All three approaches are expected to be included into the computer code and selected automatically during the calculation.

All methods for solving nonlinear equations involve the solution of many sets of linear algebraic equations. For small tasks it is possible to use simple methods based on direct solution techniques. Whenever there is sparse matrix, sparse mode elimination techniques may be successfully used. For large and super large tasks the Krylov's method expected to be applied.

Control system simulation requires the use of mathematical programming technique to get the minimization of an objective function that describes residuals. The Vicente's algorithm is suggested for solving this problem.

There are four major steps of multi-fan system simulation: (1) defining air flows and pressures for initially set control devices, (2) adjustment of positions of control devices in accordance with their set-up, (3) random selection of the positions of fume hood sashes based on their loading schedules, and (4) system performance analysis. As a result of such calculation user receives airflow, pressures, and velocities at all system sections and pressures at all system nodes. This includes workspaces, manifolds, fume hood inlet sashes, stack exhausts, and ductworks. User also receives fan operating points, position of all system control devices, and electrical energy consumption. Air system performance planed to be studied by randomly selected positions of sashes and other laboratory units. As soon as user inputs the loading schedule for each unit, computer randomizes the position of sashes/units and calculates the total usage (diversity) factor as a recommendation for optimum design.

There are four major fragments of the Bellair program: (1) preprocessor that contains of data input, verification, and printout, (2) solver that includes calculation routines, (3) postprocessor that performs verification and printout of the results, (4) populated database.

The main platform for the Bellair code is MS Windows 95/98/NT for IBM PC computers, programming language is C++, recommended compiler is Visual C++. Demo of the program planed to be installed on the

Internet. Considering inexperienced user with knowledge in duct design but not in duct simulation, the program must be user-friendly. Data input should be checked while entering. The messages should be divided into two classes: fatal and warning. They should describe the errors as well as present recommendations for correction. Full capability of inserting/copying/moving/deleting any part of the data should be implemented by using data blocking. Help screens should include examples. Graphic representation shall be used for fitting selection. The practicality of graphic input for topology and length has to be investigated. Following are the main steps of development and implementation of the Bellair program: architecture, coding and debugging, testing, data base development and population, program installation, Demo, user manual, and program support.

Abstract

The objective of this project is to develop a concept for computer modeling of multi-fan air distribution systems in research laboratories. Numerical model based on this concept will be capable of analyzing HVAC equipment, control elements, laboratory equipment, and laboratory spaces as one generalized system. Constant volume, variable volume, central supply with many exhaust air flow systems are considered. The goal is to determine the air flows and pressures in a steady-state conditions after initial data such as the position of fume hood sashes are fixed and the control sensors, controllers, and actuators have already established stability. Statistical analysis of opening fume hood sashes based on their operational schedules will allow analyzing the effect of pressurization and evaluating the efficiency of generalized air distribution system. Network simulation problem requires a solution for a large system of simultaneous nonlinear algebraic equations. Numerous methods capable of solving such equations are studied. The most promising approaches are the family of Newton Affine Invariant methods, the Tensor methods, and the Homotopy methods. If a good initial guess is known, the best choice is one of the Newton Affine Invariant methods. For a comparatively large problem, when time for solution is significant, the Tensor methods are a good choice. For the most practical cases, when either: (1) time is deficient to search for a good initial guess, (2) automation is needed for different types of technical solutions evaluation, or (3) parametric evaluation is necessary, Homotopy methods are the best. All three approaches are expected to be included into the computer code and selected automatically during the calculation. This project consists of the implementation plan for developing a multi-fan laboratory air flow simulation computer program including computer code development, data base population, testing, and support.

1. Introduction

Studies show that air conditioning systems are among the major energy consumers in laboratories. The cost of air-conditioning systems can exceed 30% of the total first cost for the entire laboratory building (Neuman and Guven, 1988). Proper system design is one of the efficient ways of reducing life-cycle cost and enhancing performance of air conditioning systems. Actual operating conditions in a laboratory served by many supply and exhaust fans vary and can result in under- or over-pressurization, imbalance, and air flow destabilization beyond design limits. Such conditions can be inefficient, unsafe, and negatively impact the work being done in the laboratories. Engineers have no practical tool to calculate actual air flows, pressures, fan operating points and the effect of system control in multi-fan laboratory air-conditioning systems.

Negative pressure confinements are very important for many laboratories dealing with harmful substances, such as nuclear materials. Positive pressure confinements are used in the "clean rooms" in order to prevent them from contamination. In some laboratories negative or positive pressure originates air infiltration or exfiltration, and creates a leakage through doorway slits and construction cracks. On the other hand, high pressure in workspace is also undesirable. If infiltration or exfiltration exceeds design limits it may result in room over- or under-pressurization, loss of pressure zoning, substantial heat and energy loss, considerable egress, dust and dirt carryover, and air flow destabilization.

The need to calculate flow distribution and internal static pressure occurs any time that an engineer is studying the effect of air flow system performance and control. The necessity for simulation appears in many HVAC designs, such as determining system operating performance, investigating system stability under different operating conditions, system retrofitting, emergency conditions and accidents, fire/smoke protection systems, pressure and air flow balancing after system modification, and equipment failure.

Due to problem complexity, this is just a first attempt to develop a practical engineering tool for multi-fan multi-lab system simulation in such wide range, which combines air flow distribution with control system and statistical analysis of equipment loading.

Quality multi-fan system design can be achieved only by using a comprehensive computer program. This project is targeted to the development and implementation of such computer program that will be used by HVAC engineers in design and retrofitting of air distribution system in research laboratories. The proposed program will allow HVAC engineer to calculate actual air flows, pressures, and fan operating points in laboratory multi-fan systems as well as the static pressure in laboratory spaces at different operating conditions where space pressurization, confinement pressure zoning, and flow/pressure stability are the most important requirements. Such computer simulation program will allow the engineer to play "what-if" scenarios that are most important in providing flexibility for laboratories with changing technology.

This computer program is planned to be distributed through the Internet free of charge and will serve companies which are involved in design, testing, and balancing of air conditioning systems for laboratories as well as the laboratory staff responsible for HVAC systems maintenance.

The purpose of the this part of the project is to define the problem, to develop the theory for a numerical modeling of multi-fan multi-lab system including determination of air pressure in the work space (typical configuration in a laboratory) and to write a plan for development, implementation, and support of computer code.

The project is divided into the following tasks:

Task One. Problem definition.
The problems of multi-fan air flow distribution was identified and classified. The majority of multi-fan problems can be described by a system of nonlinear simultaneous algebraic equations. The problem definition consists topologic and hydraulic description of the air distribution systems as well as the main requirements, types of calculation, limitations and, finally, problem formulation and main calculation flow chart. Network simulation problem is considered to be solved when the iteration process that describes steady-state hydraulic and control conditions for a given position of fume hood sashes in the network is converged. The next step of problem solution is the statistical selection of sashes positions based on their schedules. Global simulation problem is considered to be solved when the results of the statistical analyses describe the behavior of the system under different combination of sash positions.

Task Two. Analysis of existing methods.
Comparing existing numerical methods for solving multi-fan air flow distribution problems.

The existing methods under this study were compared and their strengths and shortcomings have been described and studied.

Task Three. Problem formalization.
Topology and control principles formalization is a necessary step in development of a numerical model.

Task Four. Numerical model development.
Formalized network has been used to describe the numerical model. The most beneficial methods for solving a system of nonlinear algebraic simultaneous equations have been selected and their ability to receive a fast convergence was studied.

Task Five. Software implementation plan development.
Development of an implementation plan to create, evaluate, and organize technical support of a computer program for multi-fan air flow systems simulation.

2. Problem Definition

Multiple laboratories located in the same building are usually served by a limited number of common supply systems but multiple number of exhaust systems, individual, manifolded, centralized, or combined. Such supply and exhaust systems serving rooms and corridors, fume hoods (FH), glove boxes (GB), biological safety cabinets (BSC) or other equipment, mostly with varying operational flow requirements.
For Constant Volume systems (CV) the supply and exhaust air flows once established are kept constant regardless the change of operation conditions. This is why CV systems are mostly economically inefficient and used in laboratories where supply/exhaust air must always be at the same quantities, like in many nuclear laboratories.

For Variable Volume Systems (VAV) air flow depends on operation schedule and can be reduced or sometimes even closed when equipment is in partial use or in no use.

2.1 Topology of air flow systems

Observation of various air flow systems makes it clear that they possess topological unity, according to which any network system can be seen as a version of a general system theory - graph theory. A set of nodes and arcs, where each arc is contained between two nodes is called a **graph**. Therefore, a drawing of any air flow network system is a graph.

In most cases flat graphs that corresponds to flat networks will be analyzed. However, crossing of arcs on several levels can also take place.

Every node may have one or several corresponding arcs. Their number is called the **degree of the node**. Line on a graph, which does not go through any node more than once is called a **path**.

If a line goes through each arc only once, it is called a **chain**. A cycle is a closed chain. If a chain can

connect every pair of nodes in a graph, this graph is called **linked**. A linked graph that does not contain cycles is called a **tree**. A graph containing cycles corresponds to circulation engineering systems (for example, district heating system). Any branching system is a tree. For every pair of nodes in a tree there is one and only one chain that links them.

Let us going to look at trees containing a special node, a **root**. In aerodynamic systems a root corresponds to a fan or a compressor. Of course, a rooted tree system can contain more than one root.

A chain connecting a root with a hanging node of a tree is called a **branch**. Therefore, in most common one-root trees the number of branches is equal to the number of hanging nodes - **leaves**. A node to which two arcs are adjacent is called an **elbow**. If there are more than two arcs, the node is called a **star**.

A unique property of the graphs is the flow of their nodes. Graphs are positive or negative depending on direction of their flow. Direction of flow corresponds to supply, return, and exhaust air flow systems.

All single-root trees are characterized by an important property: every group of joined arcs contains only one arc the flow in which is equal to the sum of all other arc flows with the opposite sign. Such arc is called **parent section**, and all other joined arcs are called **children sections**. For example, for a system in the Fig.1:

Node 1: children - 1, 2, 3, parent - 6;
Node 2: children - 4, 5, 6, parent - 7.

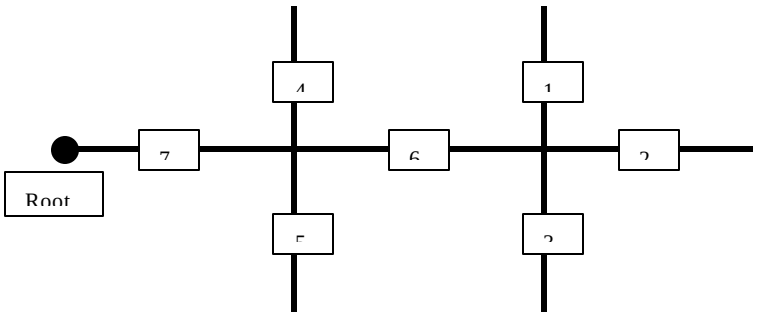

Fig.1 Tree-graph

During calculation of air flow distribution in networks it is necessary to deal either with dispersion/ gathering of the flowing medium in points of consumption, or with heat transfer. In case of heat transfer dispersion/gathering is not required, and the medium returns to the charger (fan, pump, compressor). This creates two isomorphic branched systems (supply and return), connected at the points of consumption and at the root. Such systems are called **circulation systems**. In most cases their supply and return subtrees are similar. A good example of such system is district heating and cooling.
Similar configuration takes place in air conditioning systems if supply and exhaust are considered as two tree-graphs jointed by the roots and by the terminal nodes that can be rooms, laboratories, common spaces, or corridors. For laboratories such systems can be more complicated, since they include many exhaust systems with fans from fume nodes.

Algebraic description of graph topology is done using matrices of linkages and joints.

Square matrix A=(a_{ij}) with n rows and n columns is called a matrix of linkages if a_{ij} is an element in the row i and column j that characterizes presence and sign of an arc in relation to a node i. If the flow is directed from node j to node i, then a_{ij} = 1; in all other cases a_{ij} = 0. For example, the following graph includes 2 cycles, 7 arcs, and 6 nodes (Fig.2):

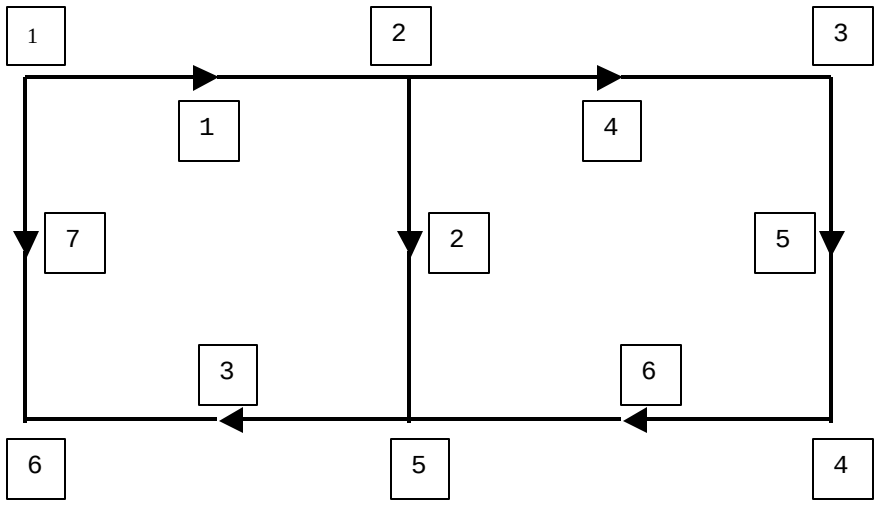

Figure 2. Two-cycle graph

Matrix $|\mathbf{A}|$ for this graph is

	1	2	3	4	5	6
1	0	0	0	0	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	0	0	1	0	0	0
5	0	1	0	1	0	0
6	1	0	0	0	1	0

Connection between arcs p_1, p_2, ... $p_÷$, and nodes x_1, x_2, ... $x_÷$ of a graph are characterized with the first matrix of incidence, $Ë_I$, where

$$ë_{ij} = \begin{cases} -1 \text{ if } p_j \text{ comes out of } x_i; \\ +1 \text{ if } p_j \text{ goes into } x_i; \\ 0 \text{ if } p_j \text{ is not joined to } x_i; \end{cases}$$

For graph on Fig.2 the matrix is:

	1	2	3	4	5	6
1	-1	1	0	0	0	0

2	0	-1	0	0	1	0
3	0	0	0	0	-1	1
4	0	-1	1	0	0	0
5	0	0	-1	1	0	0
6	0	0	0	-1	1	0
7	-1	0	0	0	0	1

The second matrix of incidence, $\ddot{E}_{II}$, can also be made. It connects arcs with cycle (cycle graphs) or arcs with branches (tree graphs).

	1	2
1	1	0
2	1	1
3	1	0
4	0	1
5	0	1
6	0	1
7	1	0

Sometime laboratory ventilation systems have unique topology that includes two symmetrical trees connected in the hanging nodes (leaves). Figure 3 represents such topology

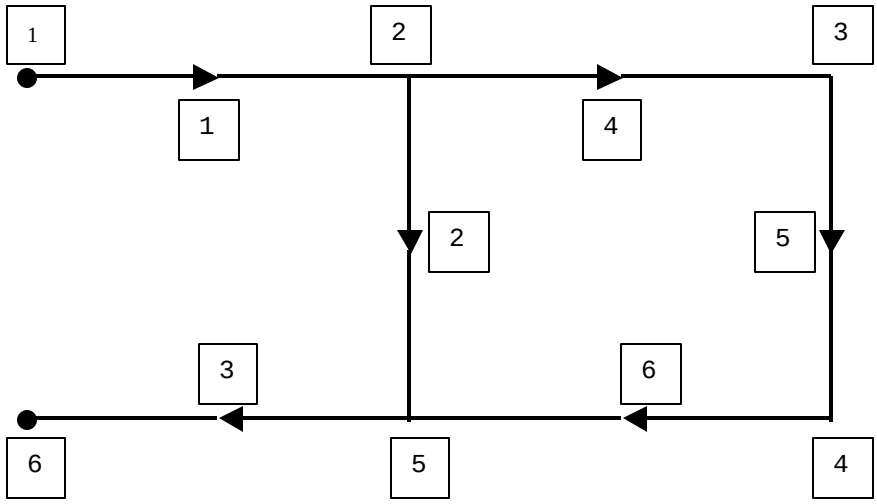

Figure 3. Six-arcs symmetric trees

Matrix $|\mathbf{A}|$ for this graph is

	1	2	3	4	5	6
1	0	0	0	0	0	0

2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	0	0	1	0	0	0
5	0	1	0	1	0	0
6	0	0	0	0	1	0

The first matrix of incidence, $\ddot{E}_I$, is:

	1	2	3	4	5	6
1	-1	1	0	0	0	0
2	0	-1	0	0	1	0
3	0	0	0	0	-1	1
4	0	-1	1	0	0	0
5	0	0	-1	1	0	0
6	0	0	0	-1	1	0

The second matrix of incidence, $\ddot{E}_{II}$, for this graph is:

	1
1	0
2	1
3	0
4	1
5	1
6	1

2.2 Hydraulics of air flow systems

2.2.1 Ducts

The Darcy-Weisbach equation for round and rectangular ducts is (ASHRAE 1997):

Round: $$\Delta P = \left(\frac{fL}{D} + \sum C \right) \frac{V^2 \rho}{2 g_c} \qquad (2.1a)$$

where:

ΔP = total pressure loss, Pa [in.WG]
f = friction factor, dimensionless
L = duct length, m [ft]
D = duct diameter, m [in]
ΣC = sum of local loss coefficients, dimensionless

V = average air velocity, m/s [fpm]
ρ = air density, kg/m^3 [lb/ft^3]
g_c = dimensional constant,(kg-m)/(N-s^2) [32.0(lbm-ft)/(lbf-s^2)]

Rectangular: $$\Delta P = \left(\frac{fL}{D_f} + \sum C \right) \frac{V^2 \rho}{2 g_c} \quad (2.1b)$$

where:
D_f = equivalent-by-friction diameter of rectangular duct, m [in]

Where the equivalent-by-friction diameter (hydraulic diameter) is

$$D_f = 2 \frac{H \times W}{H + W} \quad (2.2)$$

where:
H = duct height, m [ft]
W = duct width, m [ft]

By using the continuity equation

$$V = G / A$$

where:
G = air flow, m^3/s

And the aspect ratio for rectangular ducts (r = H/W), the following equations result:

For round ducts:

$$V = \frac{4}{\pi} G^{-2} D \quad (2.3a)$$

For rectangular ducts:

$$V = \frac{G}{H \times W} \quad (2.3b)$$

or $$V = \frac{1}{r} G^{-2} W \quad (2.3c)$$

Duct width can be interpreted in terms of an equivalent-by-velocity diameter by equating Equations 2.3a and 2.3c since Equation 2.3b can be rewritten as Equation 2.4 for round duct

$$W = 0.5\left(\frac{p}{r}\right)^{0.5} D_v \qquad (2.4)$$

Equations 2.3a and 2.3b yield the equivalent-by-velocity diameter for rectangular ducts (**D_v**)

$$D_v = 1.128\sqrt{HW} \qquad (2.5)$$

Air density ñ is a function of air temperature and pressure

$$\boldsymbol{r} = 1000\frac{\left(P_s - 0.378P_W\right)}{287.1 + \left(273.15 + T\right)} \qquad (2.6)$$

where:

P_s = static pressure in duct section, kPa [in.WG]
P_w = partial pressure of water vapor in moist air , kPa [in. WG]
T = air temperature, °C [°F]

Introducing the coefficient **ì**

Round duct: $m = fL + \sum C\ D$ (2.7)

Rectangular ducts: $m = \left(\frac{fL}{D_f} + \sum C\right) D_v$ (2.8)

Then substitute **ì** into the Darcy-Weisbach equation (2.1a and 2.1b) using Equation 2.3a to obtain the single duct pressure loss

For round duct: $\Delta P = 0.811\, g_c^{-1} m\, \boldsymbol{r} G^2 D^{-5}$ (2.9a)

For rectangular duct: $\Delta P = 0.811\, g_c^{-1} m\, \boldsymbol{r}\ G^2 D_v^{-5}$ (2.9b)

To express the diameter in terms of a pressure loss by using coefficient **ì** yields, where D is equivalent to D_v

$$D = 0.959\left(m\,\boldsymbol{r}\right)^{0.2} G^{0.4}\left(g_c\ \Delta P\right)^{-0.2} \qquad (2.10)$$

Friction coefficient in variable ì is calculated by using Colebrook's equation (1938-39).

$$\frac{1}{\sqrt{f}} = 2\log\left[\frac{12\boldsymbol{e}}{3.7D_h} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right] \qquad (2.11)$$

where:

å duct absolute roughness, m [ft]
Re = Reynolds number, dimensionless

$$\mathrm{Re} = \frac{D_h V}{\nu} \qquad (2.12)$$

where
ν = kinematics viscosity, m^2/s [ft^2/s]

Colebrook's formula cannot be solved explicitly, therefore it was simplified by Altshul-Tsal (ASHRAE 1997) as

$$f' = 0.11\left(\frac{12\varepsilon}{D_h} + \frac{68}{\mathrm{Re}}\right)^{0.25}$$

if f' ≥ 0.018 then f = f' (2.13)
if f' < 0.018 then f = 0.85 f' + 0.0028

Friction coefficient obtained by Altshul-Tsal's equation is within 1.6% of those obtained by Colebrook.

Friction coefficient for fully developed laminar flow (Re<2300) is calculated by the following formula:

$$f = 64 / \mathrm{Re} \qquad (2.14)$$

Research shows that **leakage** in an assembled duct (ÄG) can be estimated by an exponential equation. According to ASHRAE (1997) the exponent is 0.65 for turbulent flow

$$\Delta G = a_1 C_L P_s^{\,n} \qquad (2.15)$$

where:
a_1 = coefficient, 0.14×10^{-5} (1.00)

The constant C_L, called leakage class, in this equation reflects the quality of duct construction and sealing method. It is based on experimental data and exists in the range from zero for welded ducts to 110 for rectangular unsealed ducts. The average leakage class C_L for rectangular unsealed ducts is 48 (ASHRAE 1997). For laminar type leakage the flow coefficient **n** is 1. The **n** coefficient for turbulent flow leakage is in the range between 0.5 and 1.

$$\Delta G = a_1 C_L P_s^{\,0.65} \qquad (2.15a)$$

For typical duct internal static pressure is constantly changing from the static pressure at the fan to the terminal inlets/outlets due to friction, dynamic losses (fittings), and leakage. Prior to the pressure loss calculation it is unknown what the static pressure is at each duct section. Considering that duct leakage is a function of static pressure in a duct which depends on the location of fittings, Tsal/Behls/Varvak (Tsal at.el. 1998) have suggested, instead of using the sum of C-coefficient in the Darcy-Weisbach equation, to identify sections as a duct between any two fittings.

2.2.2 Fittings

Fitting resistance is calculated as a part of Darcy-Weisbach equation and can be represented by the sum of local resistance C-coefficients in a duct section unless sections are divided between fittings to obtain a high-level accuracy which is necessary for duct leakage calculation. There are a large number of tables and formulas that are used to calculate C-coefficient for fittings. The most valuable source of C-coefficient is contained in a Handbook of Hydraulic Resistance (Idelchik 1996) and the ASHRAE Duct Fitting Database (DFDB 1994).

Depending on the type of calculation, fittings can be divided into two parts: **fixed** and **variable**. Fixed fitting is the one for which its C-coefficient does not depend on unknown variables, therefore it can be calculated once at the beginning of the calculation process and does not need to be recalculated during iterations. For example, the round smooth elbow is a fixed fitting because its resistance depends only on the angle and the ratio of curved radius to the duct diameter. However, the round mittered elbow is a variable fitting because its C-coefficient depends on Reynolds number which includes air flow velocity; but this velocity is a function from air flow which is an unknown variable. Local resistance coefficients for variable fittings have to be calculated in every iteration due to the change of air flow.

There is a well-known phenomenon when C-coefficient becomes negative. Such negative C-coefficient mostly take place in junctions, converging and diverging, and interpreted as a pressure gain due to the transformation of energy instead of pressure loss. There is no "free energy" since the additional pressure always came from the another joint stream.

2.2.3 Infiltration and exfiltration

Infiltration or exfiltration to/from a room depends on pressure difference between this room and surrounding space. Air may leak through cracks and chaps in walls, ceilings, around closed doors, and electrical conduits. ASHRAE 1997 Handbook (Chapter 25, Equation 34) presents the following formula for leakage calculation

$$Q = c(\Delta P)^n \qquad (2.16)$$

where:

c = flow coefficient, (m^3/s-$Pa^{n)}$ [cfm/in.WG^n]
n = flow exponent between 0.6 and 0.7 (assumed 0.65), dimensionless

ASHRAE Handbook presents the coefficients c and n for variety of building components. The results of experimental study of these coefficients were published by Ahmed et.al. (1998).

2.2.4 Fans

The ventilation system in research laboratories can be served by a large number of fans, supply and exhaust. Computer program proposed under this project will be capable of modeling a system with many fans. Fan characteristic is the function between fan total pressure (P_f) and discharge flow (Q_f) for each rotation speed (N_f). This characteristic can be approximated either (1) by piecewise-linear way or (2) by

a number of polynomials

Rotation N_1	$Q_f = a_{11} + a_{12} P_f + a_{13} P_f^2 + a_{14} P_f^3$		
Rotation N_2	$Q_f = a_{21} + a_{22} P_f + a_{23} P_f^2 + a_{24} P_f^3$		
...	...	...	...
...	...	...	...
...	...	...	...
Rotation N_n	$Q_f = a_{n1} + a_{n2} P_f + a_{n3} P_f^2 + a_{n4} P_f^3$		

or a more complicated function (Stoecker 1975)

$$Q_f = c_1 + c_2 P_f + c_3 P_f^{\,2} + c_4 N + c_5 N^2 + c_6 P_f N + c_7 P_f^{\,2} N + c_8 P_f N^2 + c_9 P_f^{\,2} N^2 \qquad (2.17)$$

This function can be simplified by using the "Fan Laws" (ASHRAE 1996)

$$Q_{f_1} = Q_{f2} \frac{N_1}{N_2} \qquad (2.18a)$$

$$P_{f_1} = P_{f2} \left(\frac{N_1}{N_2} \right)^2 \qquad (2.18b)$$

Using these equations referenced only to one basic fan curve, the other curves for different rotation can be easily calculated.

Large number of fans manufactured and the absence of a common fan database always creates a problem for the program user who is trying to represent a fan curve by a formula. The easiest way is to input a number of representative points and approximate fan curve by lines joining these points. This can be done in a number of ways depending on selected control:

(1) Control by a damper. Input a few flows, pressures, and break horse powers (recommended not more than 8 points) and approximate the distance between them with straight lines.

(2) Control by fan rotation. Input the base fan curve in the same way as above and calculate other fan curves using "Fan Laws"

(3) Control by pitch (not practical for studied application).

(4) Controlled by inlet vane. The fan curve is a function of the vane angle presented as a per cent from fully open vanes. In general, this function is different for each fan as well as for the manufacturer; however, at the time of computer program development the study should be made how this function should be either generalized or presented in a database. The most undesirable option is contacting fan manufacturers to get the necessary data.

A very important element in calculating available discharge fan pressure is system effect factor (AMCA 1973). However, the ASHRAE is interpreting this effect as a fitting with variable C-coefficient.

2.2.5 Fume hoods

Hydraulic resistance of a fume hood (FH) depends on its types and construction. The most important three parameters:
constant or variable volume air flow,

(1) constant or variable face velocity,
(2) existence of a secondary auxiliary supply.

Auxiliary supply can be interpreted as an individual supply system that effects the room pressure (see "Problem Formalization" below). Also, a fume hood may have an individual exhaust fan and a HEPA filter. Hydraulic resistance for variable volume fume hood depends on the position of the sash and the entry air velocity.

2.2.6 Air flow systems

Relationship between air flow (G_i), pressure losses (ΔP_i) and resistances (s_i) for systems with cycles is similar to the first and the second **Kirchoff** laws.

For any node j

$$\sum_j G_i = 0, \qquad i = 1,..., p, \qquad j = 1,..., q \tag{2.19a}$$

For any c-cycle

$$\sum_c \Delta P_i = 0, \qquad i = 1,..., p, \qquad c = 1,..., k \tag{2.19b}$$

For any c-branch in a tree

$$\sum_c \Delta P_i = P_{fan} \qquad i = 1,..., p, \qquad c = 1,..., k \tag{2.19c}$$

The last two systems of equations are nonlinear (quadratic) and describes the relationship between pressures and flows for each section i (similar to Ohm's law):

$$\Delta P_i = s_i G_i^{\,2} \tag{2.20}$$

As a result, the system can be presented as a graph that includes p sections, q nodes and k cycles or branches. On this basis, a system of p equations with p unknowns can be developed. This system includes two parts. The first part is linear and is based on system of equations 2.18 for q-1 nodes. The second part describes equations 2.19 and 2.20 for k independent cycles. The number of these cycles is K=p-q+1. However, in order to develop a system of equations, one needs to know the direction of flow at each

section.

In a number of technical systems direction of flow is known. Moreover, the directions of flow can be determined by the topology of the system. In many air flow systems with central supply and central exhaust fans the direction of air flow is easy to determine. But some systems create a vicious circle, where directions of flows are defined by solving the system of equations, while the development of this system depends on the directions of flows.

There is no such problem for electrical systems, because of the sign rule. According to this rule, one can arbitrarily assign directions of currents, and then develop a system of equations based on these directions. The solution will show negative flows in some sections, which means that directions of currents in these sections need to be changed. But absolute values of currents stay unchanged. However, nonlinear aerodynamic systems do not allow determining correct directions of flows from signs of resulting flows. To make things worse, in some fittings the resistance (s_i) can be a function of flow.

At the same time, change of direction in one section causes changes of flows in all others. The general system of equations for flow distribution in quadratic aerodynamic systems with known flow directions can be represented in a matrix form vector p-dimensional space:

$$\mathbf{A\ G} = \mathbf{0} \qquad (2.21)$$

$$\mathbf{B\ S\ G\ x} = \mathbf{B\ P} \qquad (2.22)$$

where:

- **x** = vector of flows, one-column matrix.
- **A** = matrix of node coefficients. Coefficient a_i at intersections of its rows (nodes) and columns (sections) are equal to +1 if the flow is directed towards the node, and -1 if the flow is directed away from the node. Zeros are put in places where the node and the section do not intersect.
- **B** = matrix of coincidences of sections (columns) and cycles (rows). If the direction of flow coincides with the arbitrarily assigned direction of flow for the cycle, +1 is placed at the section/cycle element. In the opposite case, -1 is placed.
- **S** = square diagonal p-dimensional matrix, where the flow vector components are located along the diagonal.
- **P** = vector of pressures, one-column matrix.

Solution of systems (4) and (5) is possible through use of numerical methods and computers. But that requires knowing directions of flows in all sections, and that means that the matrix **A** is quite determined. Any linear transformation of this matrix that is connected with sign change in its columns is inadmissible.

2.3 Main requirements

Following are the main requirements for air distribution systems:

(1) **Safety.** The major safety requirement for a laboratory that contains contaminated materials is to establish room pressurization that prevents contaminants from moving into a less contaminated room

(negative pressurization). For example, there is a number of consecutive pressure zones in nuclear laboratories where negative pressure differences between zones prevent less contaminated zones from being contaminated by more contaminated zones. The pressurization effect depends on many system elements such as fans, dampers, infiltration/exfiltration, pressure control, etc. If a room has a pressure different from that of surrounding space, it creates infiltration or exfiltration that balances the difference between supply and exhaust air flow.

For a hermetically sealed room the volumes of supply and exhaust air are equal. In spite of this, the room can be under positive or negative pressure depending on the position of a "zero" pressure point in the duct system. In reality, no room can be sealed hermetically. Therefore it has infiltration or exfiltration depending on the "zero" point location.

Flow cascading is one of the main techniques used in room pressurization. There is an important limitation in this project that is the study of steady-state regime when all doorways are either closed or have airlocks.

The main safety requirement is to develop and control an air flow distribution system capable of maintaining effective laboratory pressurization to prevent contamination.

(2) **Process.** There are a number of requirements for laboratory equipment to maintain certain air velocity or air flow that effects the technological process. For example, there is a minimum-maximum velocity limit for air flow through a sash. Other example is the minimum velocity at stack exhaust. Also, necessity to maintain positive pressure in a clean room belongs to the process requirements. There is no strong boundary between safety and process requirements.

(3) **Comfort.** It is necessary to maintain the comfort conditions in the laboratories including the minimum percent of ventilation air volume, temperature, humidity, and air cleanness, as well as not exceeding the maximum noise level.

The following list is the summary of the most common requirements for air flow distribution system in laboratories. It is required for each room to maintain:

- -- Pressure zone in the minimum-maximum range,
- -- Ventilation air volume above the minimum,
- -- Face air velocity in FH and BSC sashes,
- -- Air velocity at the stack discharge,
- -- Temperature, relative humidity, and air velocity in the working zone

2.4 Types of calculations

There are two major types of calculation: optimization and simulation. Optimization takes place during system design while the simulation is applicable for retrofitting and modeling when engineering systems are analyzed for performance.

Optimization selects duct sizes and fan pressure by minimizing the system life cycle cost. Unfortunately, in reality optimization is substituted by **sizing** based on engineering judgement.

There are three types of boundary conditions associated with air system design:

(1) Boundary conditions are presented by pressures at terminals, P_i, and by fan pressure, P_{fan}. Depending on system pressures for terminal sections or fan curve may be given. This type of boundary conditions can be called the boundary conditions of the first rank.

(2) If flows are known, boundary conditions are considered to be of the second rank.

(3) If boundary conditions are presented as a mix of flows and pressures they are considered to be of the third rank.

It can be proven that in such cases boundary conditions may be presented too generally and a problem could be incorrectly determined. For example, for systems with the third rank of boundary conditions one can get a single solution, an infinite number of solutions, or no solution at all. Therefore additional data (conditions of optimization) are necessary in order to formulate the problem correctly.

Getting a solution of a **simulation** problem means determining flows, pressures, and velocities for a system with known sizes, already selected fans, established fan rotation and angles of dampers. Boundary conditions for simulation problems can require only a limited number of parameters, flows or pressures at system nodes.

The problem under this project is that of system simulation and assumes that all equipment, fans, duct sizes, and control devices have already been selected at design phase or represent an existing system.

There are strong requirements as to the number and location of control devices. A simple rule "the number of control devices has to be the same as the number of controlled parameters" works many times. It can be shown on a simple example, where a system consists of a single duct section with a fan and two dampers controlling pressure in the room. This system is overdefined and cannot work sufficiently since there is an infinite number of dampers' angles for each pressure resistance in the duct.

There are many types of air distribution systems which depends on their application and purpose: constant volume (CV), variable air volume (VAV), individual, centralized, manifolded, branched, cycled, or combined with fans connected in parallel or in series. Also, there is a variety of system elements: fans, ducts, fittings, plenums, dampers, filters, heating and cooling coils, fume hoods and biological safety cabinets, infiltration/ exfiltration resistances, more. Also, the rooms/spaces by themselves are system nodes in topological sense. Many system elements can be applicable for use under different conditions, for example, two types of fume hoods: (1) constant velocity and constant volume, and (2) constant velocity and variable volume.

2.5 Limitations

The following limitations to the problem solution will be encountered:

(1) Simulated network is limited to laboratory rooms that are served by the same central supply system. Only a single supply system with many exhaust/return systems are allowed,
(2) No economizer solution

(3) Only identical fans can be connected in parallel
(4) No dual-duct systems
(5) Only steady-state conditions are simulated
(6) Only systems that can be reduced to a flat graph are encountered
(7) All doorways are considered closed
(8) No wind effect
(9) Ambient air pressure in the surrounding rooms that are not served by the common supply system is assumed atmospheric
(10) No thermal forces like stack effects

2.6 Problem formulation

In general the goal of this project is to develop a practical technique that will be used as a tool for studying air flow and pressure in laboratories under different conditions. Proposed tool is a computer program, named BELLAIR, capable of identifying the situations when pressure confinements, process requirements, and/or comfort conditions are violated. This program will identify the cause of violating the requirements described above in part 2.3 "Main Requirements" and be used for studying preventive measures.

Following are the problem formulation for the proposed computer simulation code that includes four nested steps:

(1) **Air flow modeling**. Calculate air flows, pressures, velocities, and air leakage for each section of air distribution system under the following conditions:

-- Known topology, equipment, and sizes,
-- Known RPM (or inlet vanes opening per cent) for each fan,
-- Known angle for each damper,
-- Known open face area for each sash.

(2) **Control system modeling**. Identify the position of all control devices as a function of the set points including rotation for all fans, percentage of opening for inlet vanes, and angle of dampers. The positions of control devices have to be obtained for given open face area of fume hoods and biological safety cabinets. The goal of the control system modeling is to identify the position of actuators that satisfy following requirements:

-- Required pressure zoning,
-- Minimum ventilation rate,
-- Temperature/humidity range,
-- Air velocity range where it is required by process.

(3) **Statistical modeling**. Conduct a statistical selection of the position of each sash that depends on its working schedule. The position of the sash will be assigned randomly according to the selected time of the day and the schedule presented in input data. The idea of statistical modeling was presented by Dale Sartor (Lawrence Berkeley National Laboratory, San Francisco).

(3) **Usage (diversity) factor calculation**. It is known that all FH/BSC in many laboratories are not used simultaneously. A system usage (diversity) factor is the maximum number of exhaust devices in operation the same time (ASHRAE 1995). The computer program should create a usage (diversity) factor based on the results of the statistical analyses.

3. Mathematical and engineering methods for network simulation

3.1 Branched systems

There are a few numerical methods and computer programs for calculating flow distribution in a branched duct system. The oldest flow simulation method, called **equivalent nozzles**, was developed in Germany at the end of the 19th century by Bless (Lobaev 1959, p.79). The intent of this method is to substitute the resistance of ductwork by the equivalent resistance of a nozzle. The cross-section of the nozzle is then calculated in such a way those for the same flow the pressure losses in the system and at the nozzle are the same. Then, by computing the flow at this nozzle from the known pressure difference, the flow through each section can be calculated. The main shortcoming of this method is the need to iterate the diameters of parallel sections. The method is oriented to the quadratic law of resistance, which is seldom used for HVAC duct design.

The **unit flow** method was developed by Kamenev (1938). This method is used to calculate flow distribution in an existing system and assumes flow through the terminal section is equal to one unit of flow. Then the pressure loss per one flow unit is calculated for sections connected in parallel and in series. Flow is proportional to the number of units for each section. This method, as well as the **equivalent nozzle** method, is used in cases of quadratic law friction, which applies only when the duct velocity is greater than 70 m/s (13,700 fpm). This velocity is impractical for HVAC ducts.

The **duct characteristic** method was developed by Butakov (1949). Butakov used the old friction coefficient formula developed by Bless and substituted it into the Darcy-Weisbach equation. Then he substituted lengths, C-coefficients, and diameters for coefficients called duct characteristics and derived formulas for calculating these coefficients for sections connected in parallel and series. An important shortcoming of this method is that the use of Bless's formula results in a 20% difference between pressure loss by Colebrook (1938-1939) or Altshul (1975). This method also can be used only for ducts operating in the quadratic law friction range.

The **equivalent resistance** method was developed by Lobaev (1959) and can be used for duct sizing and system simulation. This method is almost the same as the **duct characteristic** method. Instead of using Bless's formula for calculating friction coefficients, Lobaev applied his simplified formula for metal ducts to obtain a function of hydraulics similarity that depends on friction coefficient, length, diameter, and C-coefficient.

The **steepest descent** method was applied for duct simulation by Tsal and Shor (1967) and implemented as a computer program. The objective function is the sum of square differences between fan pressure and pressure losses in branches, and the flows are the unknown variables of a system of nonlinear equations. The gradient is defined as a result of calculating a matrix of partial derivatives. The descent step is normalized at each iteration as a function of maximum gradient-vector. If the gradient is positive, it

is divided by two. The authors reported the development of a computer program for calculating the flow distribution in branches, correcting the fan operating point in the case of a change of flow, and calculating the required brake horsepower. Major applications are industrial exhaust systems conveying dust where dampers are prohibited. Tsal and Chechik (1968) developed the algorithm for flow distribution, based on the Bellman's dynamic programming method. The authors noted that this method is more difficult for implementation than the **steepest descent** method, but it has no convergence problems. The duct system is divided into a number of stages. Different pressure levels are assigned for each stage, and the method is based on a variation of flows at each pressure level. Only flows that correspond to the same pressure losses at branches are stored. Fan pressure is analyzed when the dynamic calculation process reaches the root section. The use of the dynamic programming method may be limited by available computer memory. It should be noted that Tsal's and Chechik's book was published in 1968 when computers had limited memory.

The **T-Method**, based on Bellman's Dynamic Programming ideas, has been developed by the author in cooperation with ASHRAE as a design tool for a single-root duct system simulation (Tsal, Behls, Mangel, 1990). T-Method (ASHRAE, 1997) is based on the same tee-staging idea as Dynamic Programming (Bellman 1957; Tsal and Shor 1967; Tsal and Chechik 1968). T-Method incorporates the following major procedures: (1) System condensing. Condenses the branched tree system into a single imaginary duct section with identical hydraulic characteristics and the same owning cost as the entire system (2) Selection of an operating point. Determines system flow and pressure by locating intersection of fan and system curves, (3) System expansion. Expands the condensed imaginary duct section into the original system with flow distribution. The T-method duct simulation determines the flow within each section of a duct system for known duct sizes and fan characteristics. The shortcoming of the T-Method is that it calculates tree-networks only.

A computer program called T-Duct based on T-Method was developed by R.J.Tsal (T-Duct 1994) in 1993 and successfully used for simulating duct systems with a tree-graph topology and a single fan located at the root. However, the T-Method is not capable of simulating the laboratory multi-fan systems if it is represented.

The Steepest Descent method is effectively used for modeling duct systems (Tsal and Shor 1967). Fan curve shows the relation between fan flow and fan total pressure:

$$P_{fan} = f\,(G_{fan})$$

The task is to find flow distribution in all system branches for known fan curve and known duct sizes.

Let us number sections and nodes of a system in the following manner:

$\{á_k\,(r)\}$, $k=1, ..., t_r$ are numbers of sections related to the node r,

$ÄP_á$ is pressure loss at section á,

$G_á$ is flow at section á.

The following equations take place:

1) $$\sum_{k=2}^{t_r} G_{ak(r)} = G_{a_1(r)}, \qquad r = 1,..., m \tag{3.1}$$

where:

$\alpha_1(r)$ = section number linked to the node r,
m = number of nodes in the system,
t_r = number of sections linked to the node r.

2) $$P_{fan}(G_{fan}) - \sum_{a \in M_i} \Delta P_a = 0, \qquad i = 1,..., p \tag{3.2}$$

where:

M_i = set of á indices, corresponding to sections included in branch i,
p = number of branches in the system.

Notice that ÄP is the total pressure loss, including both friction and fitting losses. When the diameters are known, $ÄP_á$ depends only on $G_{ák\,(r)}$.

On the other hand,

$$G_a = \sum_{1 \in R_a} G_i \tag{3.3}$$

where:

$R_á$ = set of branch indices that are the children of section á. If G_i (i = 1,..., p) are undefined variables, and $G_á$ is determined according to the Formula 3.3, then the Equation 3.1 will be satisfied.

Pressure loss $ÄP_á$ can be looked at as a function of G_i. Therefore, the task of flow distribution is to find a vector $\{G_j\}$, which satisfies the following system of equations:

$$P_{fan}\left(\sum_{j=1}^{p} G_j\right) - \sum_{a \in M_i} \Delta P_a\left(\{G_j\}\right) = 0, \qquad i, j = 1,..., p \tag{3.4}$$

Let us view our problem as a minimization of the function:

$$F = \sum_{i=1}^{p} R_i^{\,2}$$

Where the residuals in Equation (3.4) are:

$$R = P_{fan}\left(\sum_{j=1}^{p} G_j\right) - \sum_{a \in M_i} \Delta P_a\left(\{G_j\}\right) \tag{3.5}$$

The vector-gradient is calculated using the following formula:

$$\left\{\frac{\partial F}{\partial G_j}\right\} = \left\{2\sum R_i \frac{\partial R_i}{\partial G_j}\right\}, \qquad j = 1,..., p \tag{3.6}$$

The derivatives $\left\{\frac{\partial R_i}{\partial G_j}\right\}$ are hard to get in explicit form. Therefore, they are substituted by

finite differences:

$$\left\{\frac{\partial F_i}{\partial G_j}\right\} \approx \frac{R_i\left(G_1,..., G_j + \boldsymbol{d},..., G_p\right) - R_i\left(G_1,..., G_j,..., G_p\right)}{\boldsymbol{d}} \tag{3.7}$$

Let us look at the algorithm of the gradient descent in more detail and set the initial air flow:

$$G_1^{(0)},..., G_p^{(0)}$$

Fan curve is a part of input data. It can be approximated as a flow-pressure at a number of selected points. A partial case is considered when fan pressure P_{fan} is predetermined. Fan pressure is calculated as a function of fan curve and airflow using linear interpolation. The criterion of convergence is the following condition:

$$\frac{sign\left(P_{fan} - \sum \Delta P_a\left(\{G_j\}\right)\right)}{P_{fan}} < \max res, \qquad i, j = 1,..., p$$

where:

maxres = maximum residuals

Then derivatives are calculated as

$$\left\{\frac{\partial F_i}{\partial G_j}\right\} \approx 2\sum_{j=1}^{p} R_j \frac{R_i\left(G_1,..., G_j + \boldsymbol{d},..., G_p\right) - R_i\left(G_1,..., G_j,..., G_p\right)}{\boldsymbol{d}} \tag{3.8}$$

The descent step is determined by the maximum component of the gradient. The first step is

$$h = 0.25 \max G_i$$

Then ë (the coefficient of descent) is

$$\boldsymbol{l} = h / \max\left\{\frac{\partial F}{\partial G_j}\right\} \tag{3.9}$$

And the corrected air flows at k+1 (the next step) are:

$$G_i^{(k+1)} = G_i^{(k)} - \lambda \left\{ \frac{\partial F}{\partial G_j} \right\}, \qquad i = 1,..., p \tag{3.10}$$

At the beginning steps are large and descent is fast, but in time steps must be made smaller so that accuracy will not be lost. Therefore whenever the absolute value

$$\max\left(\frac{\partial F}{\partial G_j} \right)$$

rises, the step is divided in two.

Convergence of this method to a global minimum is guaranteed only for a concave function F. It is difficult to prove that the F function is concave due to the complicated function $\Delta P_á$. However, numerous calculations show that convergence is very good.

On the basis of the method of steepest descent R.J.Tsal created and successfully used for many years a computer program, which simulates ventilation systems (Tsal and Shor 1967). This program not only allows one to solve the problem of airflow distribution, but also corrects fan pressure in accordance with the characteristic of the fan when the flow changes and determines the power necessary.

Tsal (1998) has recently presented t-Method with Duct Leakage. The purpose of T-method simulation is to determine the flow within each section of a branched duct system of known duct sizes and fan characteristics. Incorporating duct leakage means that downstream airflow at each section is different from upstream due to air leakage through the duct walls. T-method with duct leakage incorporates the following major procedures:

System condensing. Condense the branched tree system into a single imaginary duct section with identical hydraulic characteristics. Duct leakage is simulated as an additional duct section and connected in parallel to each duct section in a duct system.

Selection of an operating point. Determine the system flow and pressure by locating the intersection of the system characteristic and the fan performance curve.

System expansion. Expand the condensed imaginary duct section into the original system with flow distribution.

Leakage at the **i**-section is simulated as an additional $\mathbf{x}_i$-section that is connected in parallel to section **i** at the node. Pressure loss for the $\mathbf{x}_i$-section is the same as for section **i** and air flow for both sections is $G=G_i+G_{xi}$. This is the main idea of incorporating duct leakage into the T-Method. Therefore the same formulas that are used by the T-Method without duct leakage are used for the T-Method with leakage incorporated. The difference is four parallel sections at each node instead of two. Condensing duct sections connected in series yields Equation 3.11 (Tsal et al. 1990, Equation 12).

$$K_{1-2} = \left(K_1^{-2} + K_2^{-2}\right)^{-0.5} \tag{3.11}$$

where:

K = hydraulic characteristic of a duct section

Condensing duct sections connected in parallel yields

$$K_{1-2} = K_1 + K_2 + K_{x_1} + K_{x_2} \tag{3.12}$$

where:

K_x = hydraulic characteristic of leakage section connected to the main section in parallel

Condensing a tee yields Equation 3.13

$$K_{1-3} = [(K_1 + K_2 + K_{x_1} + K_{x_2})^{-2} + K_3^{-2}]^{-0.5} \tag{3.13}$$

The selection and expansion procedures for T-Method duct simulation with leakage incorporated are the same as without leakage (Tsal et al. 1990).

3.2 Cycle systems

3.2.1 Methods for solving systems of nonlinear equations

The first method for water distribution in cycle systems was developed by Andriyashev and published in 1932 (Andriyashev 1932). In two years Lobachev (1934) published different technique. **Hardy Cross** from the Illinois University published his method in 1936 (Cross 1936).

For cycle network systems Martin and Peters (1963) first used the Newton-Raphson method for water supply systems. Stoecker et al. (1974) for simulating central chilled-water systems and Gregory et al. (1975) for duct systems and was later implemented into a computer code called TVENT1P. The main purpose of this program is dynamic modeling of a duct system for tornado conditions using electrical analogy for a loop system. TVENT1P uses only fixed resistance coefficients; therefore, after every iteration, the program must be interrupted. Then, C-coefficients based on output flows must be recalculated and used as input data for the next iteration. New C-coefficients have to be calculated manually for all junctions, transitions, and elbows when the C-coefficient is a function of flow, velocity, or Reynolds number. Walton (1984) successfully used the Newton method for natural airflow modeling in buildings and managed to increase the converging speed (AIRNET computer program). According to Walton (1984) a great contribution to the method used in AIRNET was provided by James Axley (1988).

The use of the Newton method was studied by Lam and Wolla (1972). They analyzed the benefits and shortcomings of the Newton method as: (1) difficulties in getting the initial guess by establishing flow directions, (2) computational difficulties in determining the Jacobian derivatives, and (3) high n^3 cost of solving the equations.

The following three methods are most suitable for solving our problem: Affine invariant Newton method, Tensor method, and Homotopy method. However, the Homotopy method should be considered as the

method of choice. Discussion about why we came to this conclusion will be presented after detailed analysis of each method. Very significant for the effective code programming is the method for approximating the Jacobian. This discussion will include different techniques for Jacobian calculations. Finally, numerical comparison between different techniques for solving linear equations will be furnished. In last five years numerical solution of linear equations gained a number of new powerful techniques. Now it is possible to reduce the computation time using such new techniques.

Numerical method for nonlinear algebraic equations has a long history. The state-of-art description presented in the excellent books by Rabinowitz (1970), Ortega (1970), Ostrowski (1966), Dennis (1983), and Shnabel (1984). A review of this subject is presented in the excellent book of Nocedal and Wrigth. (1998).

Definition of this problem: a system of nonlinear equations is mapping function F from R^n to R^n, for which a solution of x is required under conditions F(x)=0. The techniques of solving this system of equations can be interpreted using optimization techniques. In both optimization and solution for nonlinear equations Newton's method lies at the heart of many interesting algorithms. Features such as line search, trust regions, and inexact solution of the linear algebra subproblems at each iteration are very important in nonlinear equations. The most common methods for such analysis are: Newton's, Broyden's , Tensor, and Homotopy.

3.2.2 Newton's method

Let us define a linear model function M_k by taking the first two terms of the Teylor's series for Fxk

$$M_k(p) = F(x_k) + J(x_k)p \approx F(x_k + p) \tag{3.14}$$

It is assumed in Newton's method that $J(x_k)$ is nonsingular and defines as

$$p_k = -\left(J(x_k)\right)^{-1} F(x_k), x_{k+1} = x_k + p_k \tag{3.15}$$

If we assume that current iteration x_k is close to x^* and $J(x_k)$ is nonsingular convergence of Newton's method is proven (symbol * means good approximation). When Lipschitz continuity assumption is satisfied, we can prove that stronger convergence results indicate quadratic convergence. In the ideal circumstances (closeness of x_k to the solution, availability of J, exacts calculation of p_k), the basic Newton's method converges well. However these circumstances usually arrives only at the end of the solution, if ever.

Shortcomings of Newton's method include:

(1) When the starting point is remote from a solution, the basic Newton method can behave erratically. There is no guarantee that it will eventually approach a solution.
(2) The Jacobian J is required. In many circumstances it is difficult or impossible for the user to supply computer code to calculate this matrix.

(3) The Newton step p_k is usually defined by solving the linear system $J(x_k)\, p_k = -F(x_k)$. It may be too difficult to define an exact solution to this system when n is a large number. It may even be too difficult to store the coefficient matrix $J(x_k)$.

(4) The Jacobian may be singular at the root x^k

Newton's method can be modified and enhanced in various ways to gets around of the most of these problems.

Newton-Raphson method generally assures fast convergence. However, Newton-Raphson method may fail. Good explanation of this phenomenon was presented by the M.J.D.Powel (Rabinowitz, 1970).

He shows that the damped Newton-Raphson method in the form of

$$x^{k+1} = x^k + \lambda^k d^k \tag{3.16}$$

Where λ^k is calculated to prevent the estimate $x^{(k+1)}$ from being worse than the estimate $x^{(k)}$. Let use the sum of squares of residuals

$$F(x) = \sum_{i=1}^{n} [f_i(x)]^2 \tag{3.17}$$

Let us calculate $\lambda^{(k)}$ to achieve the improvement

$$F(x^{(k+1)}) < F(x^{(k)}) \tag{3.18}$$

It is possible to obtain this inequality because if $J^{(k)}$ is non-singular, algebra presents the result in the following expression

$$\left[\frac{\partial}{\partial \lambda} F(x^{(k)} + \lambda d^{(k)})\right] = -2F(x^{(k)}) < 0 \tag{3.19}$$

unless $x^{(k)}$ is already a solution of the equations. According to Powel (Rabinowitz, 1970), one recurring deficiency of this algorithm is that the successive estimates $x^{(k)}, x^{(k+1)}, \ldots, x^{(m)}, \ldots$ may converge to a point where equations are not satisfied. Also there is an obvious change to the variables that will decrease all the residuals $f_i(x)$ because the point of convergence is not the stationary point. Modification to the classical Newton-Raphson iteration, which overcomes the deficiency, is an algorithm where Equation 3.16 is replaced by the iteration

$$x^{(k+1)} = x^{(k)} + h^{(k)} \tag{3.20}$$

Where $h^{(k)}$ is the solution for the following system of linear equations

$$\sum_{j=1}^{n}\left\{m^{(k)} I_{ij}+\sum_{t=1}^{n} J_{ti}^{k} J_{tj}^{k}\right\} h_j^{(k)} = -\sum_{t=1}^{n} J_{ti}^{(k)} f_t(x^{(k)}), i = 1,2,...,n \qquad (3.21)$$

The matrix I is the unit matrix and $m^{(k)}$ is the non-negative parameter. These values are calculated to provide the inequality $F(x^{(k+1)}) < F(x^{(k)})$. Note that $m^{(k)} = 0$ is the classical iteration, and, if $m^{(k)}$ becomes large, the solution of the linear equations tends to be the value

$$h_i^{(k)} \approx -\sum_{t=1}^{n} J_{ti}^{(k)} f_t(x^{(k)}) / m^{(k)} = -\frac{1}{2}\left[\frac{\partial}{\partial x_i} F(x)\right]_{x=x^{(k)}} / m^{(k)}$$

(3.22)

showing that $h^{(k)}$ tends to a small negative multiple of the gradient of F(x) at $x = x^k$. Therefore large values of $m^{(k)}$ tend to make the iteration (3.20) similar to the classical steepest descent method applied to the function F(x). Unless $x^{(k)}$ is a stationary point of F(x), value of $m^{(k)}$ can be calculated from the required Inequality 3.18. As a result, a famous hybrid algorithm (as example, MINPACK) for nonlinear equations very similar to the Levenberg/Marquardt method has been created. The most important benefit of this approach is that it does not require explicit expressions for the derivatives, instead it uses the successive values of $f_i(x^{(k)})(i = 1,2,...,n; k = 1,2,...)$ to build up a numerical approximation to the Jacobian matrix by the technique presented in the Broyden method (see below). Revising the Jacobian approximation by this method requires only n^2 computer operations for each iteration. However if full Newton-Raphson correction is too large, the displacement from $x^{(k)}$ is based toward the steepest descent direction of F(x). This is the important feature of the Levenberg/Marquardt iteration. It is possible to find that under very mild conditions the F(x) becomes very small because $x^{(k)}$ is close to a stationary point of F(x). The famous MINPACK1 is based on hybrid algorithm. Rabinowitz (1970) presented a detailed description of the solutions.

The following are the main Powell's results.

To begin k-th iteration requires step-length Δ^k and two numbers: E and M and solving results of x^k. The step-length can be changed at each iteration, and its purpose is to restrict the length of the displacement $\left(x^{k+1} - x^k\right)$ in a way that the iteration decreases the value of F(x). By decreasing F(x) substantially the value of Δ^k is kept large. Fixed positive values are assigned to the numbers E and M before the iterations start. These multipliers govern the conditions for finishing the iterative process. It stops if F(x) is reduced to less than E, or alternatively if the gradient of F(x) is so small that the distance from x to a

solution is predicted to exceed M. Therefore E is set to a small enough value, in order the condition

$$\sum_{i=1}^{n}\left[f_i(x)\right]^2 < E \tag{3.23}$$

implies that x is close to a solution of nonlinear equations. Matrix M is usually set to an over-estimate of the distance from x^k to the solution, in order that other stopping condition is obtained only when x is close to a stationary point of F(x). The k-th iteration calculates the elements of Jacobian matrix at x^k. Then it evaluates both the full Newton-Raphson correction d^k and also the gradient g^k of F(x) at x^k. Finally, it tries to test the following inequality

$$F\left(x^k\right) \geq M \left\|g^k\right\|_2 \tag{3.24}$$

If this inequality is satisfied the iterations are completed since it is likelihood that the sequence of estimates x^k is converging not to a solution of the equations, but to a local minimum of F(x).

Theorem 1. *If the elements of the Jacobian matrices $J_{ij}^{(k)}$ are bounded and $F(x^{(k)})$ is finite, than either the iterative process is completed, because one of the conditions (3.23) or (3.24) are satisfied, or the successive estimates $x^{(1)}, x^{(2)},\ldots$ converged to a point $x^{(*)}$.*

Theorem 2. *If the function $f_i(x)$ has continuous bounded first derivatives, and Powel's algorithm is applied to solve the system of nonlinear equation F(x)=0, the algorithm converges after a finite number of iterations, due to either one of the conditions (3.23) or (3.24) being satisfied.*

Although Theorem 2 is satisfied when exact expressions for the elements of the Jacobian matrix are available, a generalization is possible when the elements of the Jacobian matrix are approximated numerically.

Theorems 1 and 2 indicate that hybrid algorithm may fail only because the functions $f_i(x), i = 1,2,\ldots,n,$ do not have bounded continuous first derivatives, or because of computer rounding errors. However, this process may terminate if the length of the predicted gradient of F(x) is less than F(x)/M, and this fact will not be acceptable to some practical cases. Because the algorithm makes frequent use of the Euclidean lengths of vectors, it is important to choose the scale of the components of x so their magnitudes are similar. This is a real deficiency, so it would be useful to include some automatic scaling. However the problem is a difficult one, because, if the equations are linear, the natural change of variables causes the surfaces F(x)=0 constant to be spherical, in which case the Newton-Raphson correction of Equation 3.20 is along the steepest descent vector of F(x). There is a danger that too much automatic scaling would

cause the hybrid method to degenerate into iteration. The radical solution of this problem is the use of affine invariant Newton's methods.

3.2.3 Broyden's method

The second type of methods, also known as quasi-Newton's methods, does not require solving the Jacobian. They construct their own approximation to the true Jacobian updating at each iteration so that it mimics the behavior of the true Jacobian over the step just taken. The approximate Jacobian is then used to compute the new search direction. In order to formalize this idea let the Jacobian approximation at iteration k be denoted be B_k. Assuming that B_k is nonsingular, define the step to the next iteration by

$$p_k = -B_k^{-1} F(x_k),\ x_{k+1} = x_k + p_k \tag{3.25}$$

Let s_k and y_k denote the differences between successive iterates and successive functions, respectively:

$$s_k = x_{k+1} - x_k,\quad y_k = F(x_{k+1}) - F(x_k) \tag{3.26}$$

According to the Taylor's theorem, s_k and y_k are related by the expression

$$y_k = \int_0^1 J(x_k + t\, s_k)\, s_k\, dt \approx J(x_{k+1})\, s_k + o(\|s_k\|) \tag{3.27}$$

The updated Jacobian approximation B_{k+1} is required to satisfy

$$y_k = B_{k+1} s_k \tag{3.28}$$

This secant condition ensures that B_{k+1} and $J(x_{k+1})$ are behave similarly along the direction s_k. However, it doesn't say anything about how B_{k+1} should behave along directions orthogonal to s_k. In fact, Equation 3.28 can be viewed as a system of n equations in n^2 unknowns (the unknowns are the components of B_{k+1}). In the scalar case n=1, Equation 3.28 defines B_{k+1} uniquely presents the well-known secant method. Otherwise, there is a subspace of matrixes B_{k+1} of dimension n(n-1) that satisfies Equation 3.28. The most successful practical algorithm is Broyden's method, for which the update formula is

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k} \tag{3.29}$$

The Broyden's update makes the smallest possible change to the Jacobian consistent with Equation 3.28. The following lemma proves this claim.

Lemma 1. *Let the matrix norms ||.|| and |||.||| be such that for any* $n \times n$ *matrices B and C and any n-vector s, the following is true*

$$\|BC\| \le \|B\|\|C\|$$

and

$$\left\| \frac{s s^T}{(s^T s)} \right\| = 1$$

Among all matrices B satisfying $B s_k = y_k$, *the matrix* B_{k+1} *defined by (3.29) minimizes*

$$\|B - B_k\|$$

The conditions of Lemma 1 is true when ||.|| and |||.||| are both the Euclidean norm, and also when ||.|| is the Frobenius norm and |||.||| is the Euclidean norm. Under certain assumptions Broyden's method converges superlinearly, that is

$$\|x_{k+1} - x^*\| = o(\|x_k - x^*\|) \tag{3.30}$$

This asymptotic rate is fast enough for most practical purposes, though not as fast as Newton's method, which converges quadratically under similar conditions. It is possible to state formal local convergence results for Newton's method and Broyden's method.

Theorem 3. *Suppose that F is continuously differentiable in an open convex set* $D \subset R^n$. *Suppose that there are an* $x^* \subset D$ *and positive scalars r and â such that F(* x^* *)=0,* $J(x^*)$ *is nonsingular with* $\|J(x^*)^{-1}\| \le b$, $B(x^*, r) \subset D$, *and J(x) is Lipshitz continuous on* $B(x^*, r)$ *with Lipshitz constant L. Then there exist å > 0 such that if* $\|x_0 - x^*\| \le e$, *the Newton sequence generated by (3.25) is well defined, convergent to* x^*, *and satisfies the following inequality*

$$\|x_{k+1} - x^*\| \le bL \|x_k - x^*\|^2$$

Roots for which $J(x^*)$ is singular are sometimes called degenerate roots. They do not satisfy the conditions of Theorem 3 and quadratic convergence cannot be expected.

3.2.4 Merit functions: nonlinear equations as optimal problem

Neither Newton's method not Broyden's method in their pure forms can guarantee to converging to a solution of F(x)=0 unless they are starting near solution. Sometimes the components of the function or Jacobian will blow up, or iterates will erratically through regions far from solutions. Eventually, an iterate might stray near enough to a root that convergence follows.

Another, more exotic kind of behavior is the cycling, where iterates move between distinct regions of the parameter space without approaching a root. What is necessary is a way to measure the "goodness" or "merit" of each point, so that we can decide whether a step is taking us to a better point or whether it needs to be modified. The most widely used merit function for nonlinear equations is the 2-norm ||F(x)|| and the closely related sum-of-squares.

$$r(x) = \frac{1}{2}\sum_{i=1}^{n} F_i^{\,2}(x) \tag{3.31}$$

Solutions of the nonlinear equations problem F(x)= 0 obviously have ||F(x)||=0 and therefore are local minima of these merit functions. Conversely, local minima of ||F|| are not necessary solutions of F(x)=0, but the strategy of seeking local minima of r is often used, since it has proved to be successful in practice.

In order to extend the convergence domain of the Newton method, some globalizations are in common use, e.g. damped Newton methods, steepest descent methods, and Levenberg-Marquadt methods. Based on the latter techniques, some state-of-the-art software has been developed, e.g. the codes from IMSL, NAG, and MINPACK. In contrast to this, the codes presented here are based on the affine invariant damped Newton techniques according to Novak (1991).

The usual local Newton techniques, within these algorithms, are combined with a special damping strategy in order to create globally convergent Newton schemes. One essential property of these algorithms is their affine invariance.

3.2.5 Global affine invariant Newton's technique

The short description of the invariant Newton's technique (Novak 1991) is presented below. The detailed description can be found in the publication by Hohmann (1993). The purpose of this technique is to solve a system of n nonlinear equations with n unknowns:

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= 0 \\ \vdots \qquad\qquad & \vdots \\ f_n(x_1,\ldots,x_n) &= 0 \end{aligned} \tag{3.32}$$

or, in the short notation

$$F(x)=0 \tag{3.33}$$

Where

F: $D \rightarrow R^n$

Where

$x=\left(x_1,\ldots,x_n\right)^T \in D \subset R^n$

Usually, some a priori information about the solution point x^* of Equation 3.33 is available in form of initial guess x_0. Besides the nonlinearity of F and the dimension n of the system, the quality of this information will strongly affect the computational complexity of solving Equation 3.33 numerically. It is one of the shortcomings of the method from the practical engineering point of view. When computer time was unavailable or very expensive, human analysis how to calculate manually a starting point for a computer implemented iterative method made good economic sense. With computer power widely available and relatively cheap, computer time is far cheaper than the time required for human analysis. In effect, the intent is to transfer intelligence from the user to the numerical algorithm. For this reason Newton method can not be used alone in practical applications.

Taking Equation 3.33 into account, the complete problem formulation may be written as

F(x)=0, $x \in R^n$

And x_0 as a given initial guess. (3.34)

First, consider the ordinary Newton method for problem Equation 3.34. Let the following equation

$$J(x) := F'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \cdots & & \cdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{3.35}$$

denote the Jacobian (n,n)- matrix, assumed to be nonsingular for all $x \in D$. Then, for a given starting point $x_0 \in D$, the ordinary Newton iteration reads

k=0,1,2,….

a) $J(x_k)\Delta x_k = -F(x_k)$

$x_{k+1} = x_k + \Delta x_k$

(3.36)

Δx_k : ordinary Newton correction.

This method is known to be quadratic convergence near the solution x^*, i.e. only a few iteration steps are necessary to generate a highly accurate numerical solution for Equation 3.34. However, the scheme described by Equations 3.36 is only locally convergent, i.e. the initial guess x_0 must be close enough to the solution x^*. To achieve convergence also for a "poor" initial guesses x_0, one may globalize Equations 3.36 by introducing a damping parameter ë . With that in mined, Equation 3.36 is extended to a damped Newton iteration k=0,1,2,….

b) $J(x_k)\Delta x_k = -F(x_k)$

$x_{k+1} = x_k + \lambda_k \Delta x_k$

(3.37)

λ_k: damping factor ($0 < \lambda_k < 1$)

Indeed, in order to create an efficient and robust method for solving Equations 3.37, this iteration must be combined with an adaptive step length control. Within such a procedure the dumping factors λ_k should be chosen in such a way, that the iterations x_k approach successively and fast the solution x^*. If the "true" convergence criterion

$$\|x_{k+1} - x^*\| < \|x_k - x^*\|, \; x_k \neq x^* \tag{3.38}$$

and the associated stopping criterion

$$\|x_{k+1} - x^*\| \leq tol \tag{3.39}$$

are computationally not available, substitute approach criteria must be introduced. Usually, such criteria are based on the definition of a so-called level function (test function). A widely used level function is given by

$$T(x) := \frac{1}{2}\|F(x)\|^2 = \frac{1}{2} F(x)^T F(x) \tag{3.40}$$

Inequalities 3.38 and 3.39 may be substituted by

$$T(x_{k+1}) < T(x_k) \tag{3.41}$$

$$\sqrt{T(x_{k+1})} \leq tol \tag{3.42}$$

The main objection to this type of criteria is that the checks of conditions 3.41 and 3.42 are made in the

"wrong" space, namely in the space of the residuals. Instead of this, the approach and the quality of iterations x_k should be checked in the space of the solution x^*. This requirement is also a consequence of the affine invariance principle, which is the leading theoretical concept of the Newton techniques.

Affine invariance. Let's introduce variable A that denote an arbitrary nonsingular (n,n) -matrix, i.e. let $A \in GL(n)$. Under this condition Equation 3.33 is equivalent to any problem described as

$$A\,F(x) = 0 \tag{3.43}$$

In other words, Equation 3.33 is invariant under the affine transformation

$$F \to G := AF,\ A \in GL(n) \tag{3.44}$$

Equivalently, Equation 3.33 is the affine invariant. This property is shared by the ordinary Newton method. Application of method described by Equations 3.36 to the transformed problem described in Equation 3.44 yields corrections

$$\Delta x_k^G = -G'(x_k)^{-1} G(x_k) = -F'(x_k)^{-1} A^{-1} AF(x_k) = -J(x_k)^{-1} F(x_k) = \Delta x_k^F \tag{3.45}$$

Hence, starting with the same initial guess $x_0^G = x_0^F$, an identical iteration sequence $\{x_k^G\}_{k=0,1,2,\ldots} = \{x_k^F\}_{k=0,1,2,\ldots}$ is generated. Consequently, for a damped Newton method of type described under condition 3.37 the affine invariance property of Equation 3.34 should carry over. Thus, an adaptive step length control algorithm for condition 3.37 must be formulated in terms of affine invariant quantities including affine invariant substitute criteria for Inequalities 3.41 and 3.42.

Damping strategy. The above mentioned outstanding choice $B := J_k^{-1}$, as well as the selection of an associated optimal damping factor λ_k for condition 3.37 is based on a substantial theoretical study. In order to characterize the nonlinearity of the problem, one may introduce an affine invariant Jacobian Lipschitz constant ω. Assume that a global constant ω exist such that

$$\left\| J(y)^{-1}[J(y) - J(x)] \right\| \le \omega \|y - x\|$$
$$x, y \in D, \omega < \infty \tag{3.46}$$

Then, with the convenient notation

$$h_k := \|\Delta x_k\| \bullet \omega,\ \overline{h}_k := h_k\, cond(BJ(x_k)) \tag{3.47}$$

following inequality holds

a) $T(x_k + \lambda \Delta x_k \mid B) \le \tau_k^2(\lambda \mid B) T(x_k \mid B)$ (3.48)

b) $\tau_k(\lambda \mid B) = 1 - \lambda + \frac{1}{2}\lambda^2 \bar{h}_k$

The result shown in Expressions 3.48 implies, that maximizing the descent means minimizing $\tau_k(\lambda \mid B)$. Straightforward calculations leads to the optimal choice

$$\lambda_k^{opt}(B) = \min\left\{1, \frac{1}{\bar{h}_k}\right\} \tag{3.49}$$

$$B_k^{opt} = J(x_k)^{-1}$$

with the extreme properties $(B \subset GL(n))$

$$a)\ \tau_k(\lambda \mid J_k^{-1}) \le \tau_k(\lambda \mid B)$$

(3.50)

$$b)\ \lambda_k^{opt}(J_k^{-1}) \ge \lambda_k^{opt}(B)$$

Insertion into (3.47) and (3.49) respectively yields a priory estimate for the damping factor Newton path. If the Newton path (from x_0 to x^*) does not exist, the damped Newton iteration will, in general, fail to converge. This situation occurs, if the solution point x^* and the initial guess x_0 are separated by a manifold with singular Jacobian. Typically, this case may arise in problems, where multiple solution points exist. In practical applications, however, different solution points x^* usually have a distinct physical interpretation. In any case, for the case of the fail of the damping method we need a method to crossover of the singular Jacobian manifold. One of such methods is the tensor method. Another solution is always to have a good initial approximation. It is a possibility of using the method discussed together with the homotopy method.

Basic algorithmic scheme. The following informal algorithm shows the basic structure of a damped affine invariant Newton iteration (including step length strategy) due to (Novak 1991). Essentially, this scheme consist of the outer Newton iteration loop and an inner step length reduction loop. This a posteriori loop is part of the outer loop and may be performed repeatedly. The Newton step comprises control of convergence and check for termination, as well as a priori estimate for the damping factor ë. Within the step length reduction loop just a refined (a posteriori) damping factor is selected and the convergence of the associated refined Newton iterate is checked again.

Input:

x_0 initial guess,

tol required accuracy,

λ_0 initial damping factor,

$\lambda_{\min}$ minimal permitted damping factor,

itmax maximum permitted number of iterations,
user routine to evaluate the nonlinear system function F(x),
user routine to evaluate the Jacobian of the system $J(x) := \frac{\partial F}{\partial x}$
standard routines for direct solution of linear equations.
Start:
k:=0
Evaluate system

$$F_k = F(x_k)$$

Newton step: evaluate Jacobian

$$J_k := J(x_k)$$

compute ordinary Newton correction

$$\Delta x_k := -J_k^{-1} F_k$$

compute a priory-damping factor

if(k > 0) $\lambda_k^0 := \min\{1, \frac{1}{[h_k^0]}\}$

where

$$[h_k^0] := \frac{\|\Delta \bar{x}_k - \Delta x_k\| \|\Delta x_k\|}{\|\Delta x_{k-1}\| \|\Delta \bar{x}_k\|} \lambda_{k-1}$$

else $\lambda_k^{(0)} := \lambda_0$
j:=0

$$\lambda := \max\{\lambda_k^0, \lambda_{\min}\}$$

a posteriori loop computes following trial iterations

$$x_{k+1}^{(j)} := x_k + \lambda \Delta x_k$$

evaluate system

$$F_{k+1}^{(j)} := F(x_{k+1}^{(j)})$$

compute simplified Newton correction

$$\Delta \bar{x}_{k+1}^{(j)} := -J_k^{-1} F_{k+1}^{(j)}$$

terminate check
exit $= \left(\|\Delta \bar{x}_{k+1}^0\| \le tol \wedge \|\Delta x_k\| \le \sqrt{tol \cdot 10} \wedge \lambda = 1\right)$

if exit: $x_{out} := x_{k+1}^0 + \Delta \bar{x}_{k+1}^0$

solution exit.
Compute a posteriori-damping factor

$$\lambda_k^{j+1} := \min\left\{1, \frac{1}{\lceil h_k^{(j+1)} \rceil}\right\}$$

where

$$\lceil h_k^{(j+1)} \rceil := \frac{2}{\lambda} \frac{\left\| \Delta \bar{x}_{k+1}^{(j)} - (1-\lambda)\Delta x_k \right\|}{\left\| \Delta x_k \right\|}$$

monotonicity check

$$\text{conv:} = \left\| \Delta \bar{x}_{k+1}^{(j)} \right\| \le \left\| \Delta x_k \right\|$$

if conv : $\Delta \bar{x}_{k+1} := \Delta \bar{x}_{k+1}^{(j)}$

$$x_{k+1} := x_{k+1}^{(j)}$$

$$F_{k+1} := F_{k+1}^{(j)}$$

$$\lambda_k = \lambda$$

k:=k+1

if(k>itmax):fail exit

proceed at Newton step

else j:=j+1

if $\left(\lambda = \lambda_{\min}\right)$: *fail* exit

$$\lambda := \min\left\{\lambda_k^{(j)}, \frac{\lambda}{2}\right\}$$

$$\lambda = \max\left\{\lambda, \lambda_{\min}\right\}$$

proceed at a posteriori loop.

In order to perform one Newton step with this scheme, the essential computational work turns out to be: one evaluation of J(x), one evaluation of F(x) and the solution of two linear systems - as long as no a posteriori step length reduction is necessary. In such a case, each reduction step requires additionally one evaluation of F and one linear system solution, but this device is activated quite rarely.

Convergence Criteria. The insight into the behavior of the damped Newton algorithm can be given by the next theoretical consideration. Let us introduce the generalized level sets

$$G(x_k \mid B) = \{x \in R^n \mid T(x \mid B) \le T(x_k \mid B)\} \tag{3.51}$$

and write the monotonicity criteria for iteration as

$$x_{k+1} \in G(x_k \mid B) \tag{3.52}$$

As long as B is not yet specified, a generalized, but natural requirement for an iterative method is that the next iteration x_{k+1} descends for all possible choices $B \in GL(n)$. Symbolically:

$$x_{k+1} \in \overline{G}(x_k)$$
$$\overline{G}(x_k) = \bigcap_{B \in GL(n)} G(x_k \mid B)$$

Under certain assumptions, the intersection $\overline{G}(x_k)$ exists and turns out to be a topological path $\bar{x} : [0,2] \to R^n$, the so called Newton path, which satisfies

$$\text{a)} \quad \begin{aligned} F(\bar{x}(s)) &= (1-s)F(x_k) \\ T(\bar{x}(s) \mid B) &= (1-s)^2 T(x_k \mid B) \end{aligned}$$

$$\text{b)} \quad \begin{aligned} \frac{d\bar{x}}{ds} &= -J(\bar{x})^{-1} F(x_k) \\ \bar{x}(0) &= x_k,\ \bar{x}(1) = x^* \end{aligned} \tag{3.53}$$

$$\text{c)} \quad \frac{d\bar{x}}{ds}\Big|_{s=0} = -J(x_k)^{-1} F(x_k) \equiv \Delta x_k$$

The constructed Newton path $\bar{x}$ is outstanding in the respect that all level functions T(x|B) decrease along $\bar{x}$. Geometrically the damped Newton step in x_k continues along the tangent of the Newton path $\overline{G}(x_k)$ with an appropriate length and at the next iterate x_{k+1}, the next Newton path $\overline{G}(x_{k+1})$ will be chosen for orientation towards x^*. These considerations show the limit of the affine invariant Newton techniques. If the Newton path from x_0 to x^* does not exist, the damped Newton iteration will, in general, fail to converge. This situation occurs, if the solution point x^* and the initial guess x_0 are separated by the manifold with singular Jacobian.
Typically, this case may arise in problems, where multiple solution points exist.

Theorem 4. *Let $F : D \subset X \to X$ be a continuously Frechet-differentiable mapping, such that F'(x) is continuously invertible for all $x \in X$. Moreover, it is required for F' to meet the following affine invariant Lipschitz condition*
$$\left\| F'(y)^{-1}(F'(x + t(y-x)) - F'(x))(y-x) \right\| \le tw \left\| y - x \right\|^2$$

for all x,y $\in D, t \in [0,1]$ *and some ù>0. If the initial guess* x_0 *satisfies*

$$w\|\Delta x_0\| = w\|F'(x_0)^{-1} F(x_0)\| < 2$$

and the Newton iteration x_k *stays in the domain D, then the iteration converges to a solution x of F(x)=0. The convergence is quadratic in the sense that*

$$\|\Delta x_{k+1}\| \le C\|\Delta x_k\|^2$$

for some constant C > 0.

Conclusion. The method discussed above is a computationally very effective for good initial guess. For practical applications this method must be used with numerical approach giving a good initial guess to the solution.

3.2.6 Tensor methods

Tenzor methods (Schnabel 1984) are a class of general-purpose methods for solving systems of nonlinear equations. They are especially intended to solve problems where Jacobian matrix is singular or poorly conditioned, while remaining at least as efficient as standard methods on nonsingular problems. Their distinguishing feature that they base each iteration on the simple second order term. This term allows it to interpolate more nonlinear function then standard linear model based methods without significantly increasing the cost of forming, storing or solving the model. There are two types of tensor methods, derivative that calculate an analytic or finite difference Jacobian at each iteration, and secant that avoid Jacobian evaluations. Both are require no more storage or arithmetic operations per iteration than standard linear model based methods. The attention to this approach growing after incorporating in the work (Bouricha 1992).

The main goal of tensor methods is to provide general purpose methods that have rapid convergence even when F'(x) is singular. In addition, the methods should not experience any special difficulty when J_c is singular or ill conditioned.

Tensor methods are based on expanding the linear model of F(x) around x_c to the quadratic model.

$$M_T(x_c + d) = F(x_c) + J_c d + \frac{1}{2} T_c dd \qquad (3.54)$$

Where $T_c \in R^{n \times n \times n}$ *and* J_c is $F'(x_c)$ or a secant approximation to it. The three-dimensional object T_c often is referred to as a tensor. The tensor term is selected so that the model interpolates a very small number, p, of function values from previous iterations. This results in T_c being a rank p tensor, which is crucial to the efficiency of the tensor method. After the model described by the Expression 3.54 is formed, the problem

find $d \in R^n$ that minimizes $\left\| M(x_c + d) \right\|_2$ (3.55)

is solved; that is, at each iteration of tensor methods, a minimizer of the model is used if no root exist. The tensor method requires no more derivative or function information per iteration than Newton's method, and its storage requirement and arithmetic cost per iteration are not appreciably more than for Newton's method. Bouricha (1992) shows that methods based on Expression 3.54 have good theoretical properties and good computational performance. Theoretically, tensor methods converge at least as quickly as Newton's method on nonsingular problems and have been shown to have 3-step Q-order 1.5 convergence on problems where the Jacobian has rank n-1 at the solution, whereas Newton's method is linearly convergent with constant 1/2 on such problems. The improvement by the tensor method over the standard method is substantial, averaging about 19% in iterations and 41% in function evaluations on problems solved successfully by both methods (Bouaricha 1992). Furthermore, the tensor method solves a considerable number of problems that the standard method does not, and the reverse virtually never is the case. The most difficult and expensive part of the tensor method is solving the quadratic model (Expression 3.54) efficiently.

Derivative tensor methods

Forming the tensor model. The first step in deriving a method based on Expression 3.54 is to choose the second order term T_c. The second order term constructed by asking the model to interpolate additional values of the function F(x) that have already been computed by the algorithm. In particular, model asked to satisfy

$$F(x_k) = F(x_c) + F'(x_c)s_k + \frac{1}{2}T_c e_k e_k, \; k = 1,2,..., p \qquad (3.56)$$

Where

$e_k = x_k - x_c$, k=1,2,…,p, and $x_1, ..., x_p$ are some set of p past iterates that need not be consecutive.

For the Equations 3.56 to be consistent, the past points $\{x_k\}$ must be selected so that the set of directions $\{s_k\}$ is linearly independent. In fact a far more restrictive condition enforced. Matrix x_{-1} always predetermined to the most recent iterate. Each remaining past iterate included in the set of points to be interpolated if the step from it to x_{-c} makes an angle of at least è degrees with the subspace spanned be the steps to the already selected more recent iterates. Here è is some fixed angle between 20 and 45 degrees. In addition, at most $\sqrt{n}$ past iterates are considered. This procedure for selecting past iterates to interpolate is implemented easily using a modified Gram - Schmidt algorithm, and requires about n^2 multiplication and additions.

Equations 3.56 are a set of $n \times p \le n^{1.5}$ linear equations in the n^3 unknowns comprising T_c. Thus T_c is undetermined, following the standard practice in secant methods for nonlinear equations and optimization,

choose T_c to be the solution to

$$\min \|T_c\|_F \tag{3.57}$$

$$T_c \in R^{n\times n\times n}$$
$$subject\ to\ T_c e_{-k} e_{-k} = t_{-k} = 2(F(x_{-k}) - F(x_{-c}) - F'(x_{-c}) e_{-k}), k = 1,2,..., p,$$

Once the tensor Model 3.56 is formed, a root of the tensor model is found. It is possible that no root exists; in this case a least squares solution of the model is found instead. Schnabel and Frank (1984) show that the solution to Expression 3.57 can be reduced to the solution of q quadratic equations with p unknowns plus the solution of n-q linear equations with n-p unknowns. In the case the main steps of the algorithm used to solve Expression 3.57 are follow:

1. An orthogonal transformation of the variable space is used to cause the n equations with n unknowns to be linear in n-p variables, $d_1 \in R^{n-p}$ and quadratic only in the remaining p variables, $d_2 \in R^p$.
2. An orthogonal transformation of the equations is used to eliminate the n-p transformed linear variables from n-q of the equations. The result is a system of q quadratic equations in the p unknowns, d_2, plus a system of n-q equations in all the variables that is linear in n-p unknowns d_1.
3. A nonlinear unconstrained optimization software package is used to minimize the l_2 norm of the q quadratic equations with p unknowns d_2.
4. The system of n-q linear equations that is linear in the remaining n-p unknowns is solved for d_1

An advantage of this algorithm is that it efficiently and stable solves (3.56), whether or not the tensor model has a root or the Jacobian is nonsingular. The total cost of the above algorithm is the $O(n^3)$ cost of Newton's method plus at most an additional cost of $O(n^{2.5})$ arithmetic operations .

An iteration of the tensor method is summarized in the algorithm below. For more details on tensor methods see Bouricha (1992).

Algorithm of the iteration of the tensor method for dense nonlinear equations.

Presenting n, current iterate: x_c, $F(x_c)$

1. Calculate $F'(x_c)$ and check if it is permissible to stop. If not, follow to the next step.
2. Select the past points to use in the tensor model from among the $\sqrt{n}$ most recent points.
3. Calculate the second-order term of the tensor model, T_c, so that the tensor model interpolates F(x) at all the points selected in Step 2.

4. Find the root of the tensor model, or its minimizer (in the l_2 norm) if it has no real root.
5. Select the next iterate x using either a line search global strategy or a two-dimensional trust region method.
6. Set $x_c - x,\ F(x_c) - F(x)$; return to step 1.

The procedure for solving the tensor model in the dense case, however, does not adapt to large sparse problems. The first step of this process, the orthogonal transformation of the variable space, is crucial to this approach.

Methods for approximation the Jacobian. For the majority of the practical problems it is not possible to express the Jacobian explicitly, and in these cases it is necessary to use an approximation B to the Jacobian. The most obvious one can be obtained by using a forward-difference formula to approximate the partial derivative.

Two possibilities for dealing with this situation include the use of automatic differentiation and numerical approximation of the Jacobian by finite differences. Automatic differentiation has the advantage of producing exact derivatives, which can be essential to the convergence of Newton's method in some situations. The practicality of automatic differentiation tools has been demonstrated on a wide range of applications. They can still require a certain amount of user intervention and can be unwieldy. For this reason we decided to use the second alternative, approximation of the Jacobian by finite differences. Solving the tensor model when the Jacobian is sparse.

Implementation of tensor methods for sparse nonlinear equations. The main invention of Bouricha
(1992) to the tensor methods development is the construction of an efficient algorithm for finding a root of the tensor model when Jacobian matrix is large and sparse. That is:
Find $d \in R^n$ such that

$$M(x_c + d) = F(x_c) + F'(x_c)d + \frac{1}{2}\sum_{k=1}^{p} a_k \left\{ d^T s_k \right\}^2 = 0 \qquad (3.58)$$

where $F'(x_c)$ is large and sparse. Bouricha reduced solution of this equation to the solution of a system of p quadratic equations with p unknowns, plus the solution of p+1 systems of linear equations that all involve the same matrix. This matrix is either J(x_c), if it is nonsingular and well conditioned, or J(x_c) augmented by p dense rows and columns if J(x_c) is singular or ill-conditioned. Bouricha's (1992) results are presented below. The algorithm based on the solution of the generalization of Expression 3.54

$$\text{Find } d \in R^n \text{ that minimizes } \left\| M(x_c + d) \right\|_2 \qquad (3.59)$$

Let's the Jacobian matrix is nonsingular and the tensor model has a root. Multiplying Expression 3.59 by $s_i^T J^{-1}, i = 1,..., p$ gives the p quadratic equations in the p unknowns $b_i = s_i^T d$,

$$s_i^T J^{-1} F + \beta_i + \frac{1}{2}\sum_{k=1}^{p}\left(s_i^T J^{-1}\alpha_k\right)\beta_k^2 = 0, \text{ i} = 1,..., \text{p} \qquad (3.60)$$

These equations can be solved for $\beta_i, i = 1,..., p,$ and then from (3.58) the equation

$$F + Jd + \frac{1}{2}\sum_{k=1}^{p}\alpha_k \beta_k^2 = 0 \qquad (3.61)$$

can be solved for d. The entire process required the solution of p+1 systems of linear equations in the matrix J to compute $J^{-1}F$ and $J^{-1}\alpha_k$ k=1,…,p. and the solution of the small system of quadratics (3.61).

Solving the sparse tensor model when Jacobian is nonsingular. This task handled by considering the equivalent minimization problem to Equation 3.58,

$$\min_{d\in R^n}\left|\left|QM(x_c+d)\right|\right|_2 \qquad (3.62)$$

where Q is an $n\times n$ orthogonal matrix that has the structure

$$Q = \begin{bmatrix} U^T \\ Z^T \end{bmatrix}$$

with

$U \in R^{n*p}; U = J^{-T} S\left[S^T\left(J^T J\right)^{-1} S\right]^{-1/2}$, S an (n×p) matrix

whose columns are $s_i, , i = 1,..., p$

$Z \in R^{n\times(n-p)}$ is an orthonormal basis for the orthogonal complement of the subspace spanned by the columns of $J^{-1}S$.

Bouricha presents the following algorithm for the sparse tensor model solution when J is nonsingular:

Let $J \in R^{n\times n}$ be sparse, $F \in R^n, S, A \in R^{n\times p}$.

1. Form the q(â) Equations (3.62) by calculating $J^T S$ as follows: factor J and solve $J^T y_i = s_i, i = 1,..., p.$

2. Form the positive definite matrix $W \in R^{p\times p}$, where $W_{ij} = \left\{s_i^T\left(J^T J\right)^{-1} s_j\right\}, 1 \le i, j \le p,$ as follows:

$$W_{ij} = \left(J^{-T} s_i\right)^T \left(J^{-T} s_i\right) = y_i^T y_i$$

3. Perform Cholesky's decomposition of W (i.e. W= $L L^T$) resulting in $L \in R^{p \times p}$, a lower triangular matrix.
4. Use an unconstrained minimization software package, to solve

$$\min_{b \in R^p} \left\| L^{-1} q(\boldsymbol{b}) \right\|_2^2$$

or to solve this equation in closed form if p=1.
5. Substitute the values of â and q(â) into

$$d = -J^{-1}\left(F + \frac{1}{2} A \boldsymbol{b}^2 - J^{-T} S W^{-1} q(\boldsymbol{b})\right)$$

to obtain the tensor step d. This involves one additional solution, since the factorization of J is already calculated.

The total cost of this process is the factorization of the sparse matrix J, p+1 back solution using this factorization, the unconstrained minimization of a function of p variables, and some lower-order (O(n)) costs .

Solving the sparse tensor model when the Jacobian is rank deficient. If the Jacobian matrix is rank deficient Bouricha shows that it is possible to solve the tensor model by building a calculation process just described above. The basis of this approach is to transform the tensor model given in Expression 3.56 as follows.

Algorithm. Solving the Sparse Tensor Model

Let $J \in R^{n \times n}$ be sparse, A, S $\in R^{n \times p}$.

1. Form the matrix A_{D_b}, where d is the step computed in the previous iteration, $\boldsymbol{b} = S^t d$, and $D_b = diag(\boldsymbol{b})$. Then construct the augmented matrix $M \in R^{(n+p)\times(n+p)}$ as follows

$$\text{M} = \begin{bmatrix} J & A_{D_b} \\ S^T & -I \end{bmatrix} \tag{3.63}$$

2. Begin the factorization of M, pivoting in rows and columns n+1,…,n+p only if J is (numerically) singular. If J is nonsingular, perform algorithm for the nonsingular case on the tensor model (Expression

3.58).

3. If J is singular but M is nonsingular, than perform algorithm for nonsingular case on the tensor Model

$$M(x_c+\boldsymbol{d}) = F(x_c)+J(x_c)\boldsymbol{d}+\frac{1}{2}A\left\{S^T\boldsymbol{d}\right\}^2, \tag{3.64}$$

where $F(x_c) = F(x_c)+J(x_c)d+\frac{1}{2}A\left\{S^T d\right\}^2$

and $J(x_c) = J(x_c)+A_{D_b}S^T$, and any required value of the form x= $J^{-1}b$ or x= $J^{-T}b$ is found by solving the augmented system with Matrix 3.63 or the analogous transposed system for x. Then set d=$\overline{d}+\boldsymbol{d}$.

4. If M is singular, use the singular Newton step calculated in the next section, instead of tensor step.

The arithmetic cost per iteration of this algorithm is the cost of a sparse matrix factorization of the Jacobian J or the augmented matrix M plus the costs for p+1 back solves plus the O(p^4) cost of using minimization package for solving q(â) equations if p>1, plus some O(n) costs. Thus the main additional cost in relation to Newton's method again is p additional forward and back solves per iteration.

Solving the Newton model along with the sparse tensor model. If the Jacobian matrix J is nonsingular, then the calculation of the tensor step described above produces a sparse LU factorization of J. In this case, the Newton step is simply found by performing one additional pair of triangular matrix, solves the system

$$Jd=-F \tag{3.65}$$

That is, since

$$J = P_1^T LU P_2^T \tag{3.66}$$

where $L \in R^{n\times m}$ is unit lower triangular matrix, and P_1 and P_2 are row and column permutation matrices. First solve

$$Ly=c \tag{3.67}$$

for y, where y=$U P_2^T d$ and c=- $P_1 F$. Then solve

$$Uz = y \tag{3.68}$$

for z, where z = $P_2^T d$. Finally, d= $P_2 z$. Otherwise the matrix J is singular, so Equation 3.65 is either zero or an infinite number of solutions. Therefore, it is necessary to solve the least squares problem

$$\min_{d \in R^n} \left\| Jd + F \right\|_2 \tag{3.69}$$

The method that is possible to solve this problem is an extension of the method of Peters and Wilkinson (1970) that was suggested by Bjorck and Duff (1980).

Implementation of tensor methods for sparse nonlinear equations. The global strategy that is used in this implementation is a standard line search, because of its greater simplicity and because the two-dimensional trust region method requires two additional matrix-vector multiplication involving the Jacobian matrix.

Algorithm. Given current iterate $x_c, F(x_c)$.

1. Calculate J=F'(x_c) and check whether to stop. If not, follow to the next step:
2. Form the second-order term of the tensor model, T_c, so that the tensor model interpolates F(x) at the most recent past point (i.e., p=1).
3. Factorize J using the sparse matrix software package.
4. If J has full rank, then perform algorithm for nonsingular Jacobian on the tensor model $M(x_c + d) = F(x_c) + Jd + \frac{1}{2}\sum_{k=1}^{p} \alpha_k \left(d^T s_k\right)^2$, to compute the tensor step d_t and go to Step 6. Else:

4.1 Augment J by adding p rows and columns as follows (in this implementation, p=1). In general, column k of A=a_k, column k of S=s_k, and $D_b = diag(s_k^T d)$, where d is the step computed in the previous iteration.

$$\text{M}=\begin{bmatrix} J & AD_b \\ S^T & I \end{bmatrix} \tag{3.69a}$$

4.2 Complete the factorization of the augmented matrix M as follows. Let r denote the rank of J.

4.2.1 Update the lower left rectangular p*r submatrix, and the upper right rectangular r*p submatrix of the augmented matrix M, using the multipliers stored in the L factor of the LU factorization of J.

4.2.2 Factor the lower right square (n-r+p)(n-r+p) submatrix of the augmented Matrix 3.69a using the sparse matrix software package.

4.2.3 Update the factorization of the entire augmented Matrix 3.69a by combining the LU factorization of the submatrix in Step 4.2.2, the updated submatrices in Step 4.2.1, and the LU factorization of J into one LU factorization of the augmented Matrix 3.69a.

5. If J was singular but the augmented matrix M has full rank, then perform algorithm on the tensor model

$$M(x_c + d) = F(x_c) + Jd + \frac{1}{2} A\{S^T d\}^2,$$

where

$$F(x_c) = F(x_c) + Jd + \frac{1}{2} A\{S^T d\}^2, J = J + A D_b S^T,$$

And d is the step computed in the previous iteration, to compute the step d. Then set $d_t = \bar{d} + d$ and go to Step 6. Else:

5.1 Calculate the Newton step d_n from the LU factorization of J by Bjorck and Duff (1980) method to find some solution to $\min_{d \in R^n} \|Jd + F\|_2$

5.2 Select the next iterates x_+ using line search algorithm, where d_n is the search direction, and jump to Step 7.

6. Select the next iterate x_+ using a line search global strategy as follows:

6.1 If $x_c + d_t$ is acceptable, than establish $x_+ = x_c + d_t$ and jump to Step 7. Else

6.2 Calculate the Newton step d_n from the LU factorization of J (or as in Step 5.1 if J is singular). Then calculate $x_+^n = x_c + \lambda d_n$ for some ë>0, using algorithm for quadratic backtracking line search. If the tensor step is a descent direction, then calculate $x_+^t = x_c + \lambda d_t$ for some ë>0, using algorithm for quadratic backtracking line search.

6.3 If $\left\|F(x_+^n)\right\|_2 > \left\|F(x_+^t)\right\|_2$, then $x_+ - x_+^t$, else $x_+ - x_n^+$

7. Set $x_c - x_+, F(x_c) - F(x_+)$. Return to step 1.

The sparse tensor code terminates successfully if the relative size of $(x_+ - x_c)$ is less than macheps$^{2/3}$, or $\|F(x_+)\|$ is less than macheps$^{2/3}$; it terminates unsuccessfully if the iteration limit is exceeded. If the last global step fails to locate a point lower than x_c in the line search global strategy, or the relative size of $J(x_+)^T F(x_+)$ is less than macheps$^{1/3}$, the method stops and reports this condition; this may indicate either success or failure.

Conclusion. The tensor method more robust then damped Newton's. The tensor method is a special case of the regularization operator approach. (Furi 1969, Tihonov 1986). Theoretically it is possible to find situation where tensor method will be unsuccessful because the regularization operator used sometimes can loose the compactification property. The detailed description of the methods for the ill-posed problems solution and analysis of the regularization operator properties can be found in publication by Tihonov

(1986). Method can be used by experienced users for solving practical problems but some probability exists that method will fall. Inexperienced user needs more robust approach.

3.2.7 Methods based on homotopy

This method is very robust and capable of achieving global convergence at high probability.

Globally convergent homotopy methods. It was mentioned above that the Newton-based methods suffer from a major shortcoming. They are not guaranteed to converge to a solution of F(x)=0 but only, at best, a local minimum of $\|F(x)\|$. There are algorithms for finding zeros or fixed points of nonlinear systems of equations that are globally convergent for almost all starting points, i.e. with probability one. The essence of all such algorithms is that the construction of an appropriate homotopy map and then tracking some smooth curve in the zero set of this homotopy map.

Homotopies are a traditional part of topology, and have found significant application in nonlinear functional analysis and differential geometry. The concepts of homotopy maps, continuation, incremental loading, and invariant imbedding are widely used and intertwined. Let Homotopy methods be the generic terms, including continuation, parameter continuation, incremental loading, displacement incrementation, invariant imbedding, and continuous Newton methods.

Parameter continuation is well established technique in numerical analysis, with the basic idea being to solve a series of problems as some parameter is slowly varied, using a locally convergent iterative technique for each problem, and the solution of the previous problem as the starting point for the current problem). Similar techniques in engineering are known as incremental loading and displacement incrementation. For the most part all these methods have rather restrictive hypotheses and the connection with topology had not been made

A fundamental breakthrough occurred with the truly globally convergent simple fixed point algorithms. These algorithms were grounded in topology, constructive, potentially extremely powerful, but horribly inefficient in their early forms.

Another significant advantage was the differential equation formulation of continuation, proposed in various forms by Rheinboldt (1982). Although the underlying homotopy map in these differential equation forms may have been the same as in "classical" continuation, it is important to realize that the algorithms and implementations are fundamentally different. Despite considerable success on practical problems and a large amount of supporting theory, all these homotopy methods suffered from a fatal flaw. A Jacobian matrix somewhere could become singular, and the computer implementation would either experience great difficulty or the method would fail completely.

The next advance was the development in 1976 by S.N.Chow (Chow 1978) of probability one homotopy methods. The thorn of singular Jacobian matrices was finally removed, since these methods were specifically constructed to not have any singular points. The phrase "probability one" refers to the supporting theory, which says that for almost all choices of some parameter vector involved into the homotopy map, there are no singular points and the method is globally convergent. Again, while globally convergent probability one homotopy algorithms may have a superficial resemblance to earlier homotopy

algorithms, it is important to note that the philosophy in fundamentally new. Furthermore, because of this philosophical difference, there are subtle differences in mathematical software for probability one homotopy algorithms from the other continuation methods.

Algorithms for dense Jacobian matrices. Let E^p denote p-dimensional real Euclidean space. The following four lemmas from Watson (1985) are the basis for the homotopy algorithms.

Lemma 2. *Let g:* $E^p \to E^p$ *be a* C^2 *map,* $a \in E^p$, *and define* $\rho_a : [0,1] \times E^p \to E^p$, *by*

$$\rho_a(\lambda, y) = \lambda g(y) + (1-\lambda)(y-a)$$

Then for almost all $a \in E^p$ *there is a zero curve ã of* ρ_a *emanating from (0,a) along which the Jacobian matrix D* $\rho_a(\lambda, y)$ *has full rank.*

Lemma 3. *If the zero curve ã in Lemma 2 is bounded, it has an accumulation point (1,y) where g(y)=0. Furthermore, if Dg(y) is nonsingular, then ã has finite arc length.*

Lemma 4. *Let* $F : E^p \to E^p$ *be a* C^2 *map such that for some r > 0, xF(x) 0 whenever* $\|x\| = r$. *Then F has a zero in* $\{x \in E^p \mid \|x\| \le r\}$, *and for almost all* $a \in E^p, \|a\| < r$, *there is a zero curve ã of*

$$\rho_a(\lambda, x) = \lambda F(x) + (1-\lambda)(x-a),$$

along which the Jacobian matrix $D\,\rho_a(\lambda, x)$ *has full rank, emanating from (0,á) and reaching a zero x of F at ë=1.*

Furthermore, ã has finite arc length if DF(x) is nonsingular. Lemma 4 is a special case of the following more general lemma.

Lemma 5. *Let* $F : E^p \to E^p$ be a C^2 *map such that for some r > 0 and r1 > 0, F(x) and x-a do not point in opposite directions for* $\|x\| = r, \|a\| < r1$. *Then F has a zero in* $\{x \in E^p \mid \|x\| \le r\}$, *and for almost all* $a \in E^p, \|a\| < r1$, *there is a zero curve ã of*

$$\rho_a(\lambda, x) = \lambda F(x) + (1-\lambda)(x-a)$$

along which the Jacobian matrix $D\,\rho_a(\lambda, x)$ *has full rank, emanating from (0,á) and reaching a zero x of F at ë=1.*

Furthermore, ã has finite arc length if DF(x) is nonsingular. The general idea of the algorithm is apparent from the Lemmas: just follow the zero curve ã emanating from (0,W) until a zero Y of F(Y) is reached (at ë=1). The homotopy map is

$$\rho w(\lambda, Y) = \lambda F(Y) + (1-\lambda)(Y - W) \tag{3.70}$$

that has the same form as a standard continuation or embedding mapping. However, there are two crucial differences. In standard continuation, the embedding parameter ë increases monotonically from 0 to 1 as the trivial problem Y-W=0 is continuously deformed to the problem F(Y)=0. The present homotopy method permits ë to both increase and decrease along ã with no adverse effect. This turning point presents no special difficulties. The second important difference is that there are no any "singular points" that afflict standard continuation methods. The way along which zero curve ã of ñw is followed, and the full rank of $D\rho_w$ along ã, guarantees this. According to Lemma 2, ã cannot just "stop" at an interior point of $[0,1]\times E^p$

The homotopy map ρ_a exists from (p+1)-dimensional space to p-dimensional space. This mismatch of dimensions is a double-edged sword since the extra dimension permits "lifting" the original problem to obtain a problem with no singularities and a full rank Jacobian matrix, and that can be solved from an arbitrary starting point. However, the Jacobian matrix

$$D\rho_a(\lambda, x) = \left[D_\lambda \rho_a(\lambda, x) \; D_x \rho_a(\lambda, x) \right] \tag{3.71}$$

of ρ_a is rectangular (p(P+1)), and this is an essential aspect of the lifted problem. By contrast, the Hessian matrices (Rabinowitz 70)) in nonlinear optimization and the Jacobian matrices in locally convergent algorithms for nonlinear systems of equations are always square. The p(p+1) shape adds a combinatorial aspect to the numerical linear algebra, and this subtle difference is important in computer implementations.

The zero curve ã of the homotopy map can be tracking by three primary algorithmic approaches:

1) an ODE-based algorithm
2) a predictor corrector algorithm
3) a version of Rheinboldt's linear predictor, quasi-Newton corrector algorithm (an augmented Jacobian matrix method).

ODE-based algorithm. The detailed description of this algorithm has been published by Watson (1986). A short description is presented below.

Assuming that F(Y) is C^2 and W is such that Lemma 2 holds, the zero curve ν is C^1 and can be parameterized by arc length s. Thus $\lambda = \lambda(s), Y = Y(s)$ along ã and

$$\rho_w\left(\lambda(s), Y(s)\right) = 0 \tag{3.72}$$

identically in s. Therefore

$$\frac{d}{ds}\,\boldsymbol{r}_w\big(\boldsymbol{l}(s),Y(s)\big)=D\,\boldsymbol{r}_w\big(\boldsymbol{l}(s),Y(s)\big)\begin{pmatrix}\dfrac{\partial \boldsymbol{l}}{\partial s}\\[2mm]\dfrac{\partial Y}{\partial s}\end{pmatrix}=0 \qquad (3.73)$$

$$\left\|\left(\frac{d\boldsymbol{l}}{ds},\frac{dY}{ds}\right)\right\|_2=1$$

assuming ëY(0)=W

the zero curve í is the trajectory of the initial value problem (3.72)-(3.73). When ë(s)=1, the corresponding Y(s) is a zero of F(Y). Typical ODE software requires (dë/ds,dY/ds) explicitly. One of ways is to use Algebraic Differential software (Petsold 1989). Another way, established by Watson (1986) , is calculation (dë/ds,dY/ds) as a unit tangent vector to the zero curve í by finding a one-dimensional kernel of the p*(p+1) Jacobian matrix

$$D\,\boldsymbol{r}_w(\boldsymbol{l}(s),Y(s))$$

that has full rank by Lemma 2. The crucial observation is that the last p columns of $D\,\boldsymbol{r}_w$, corresponding to $D_y\,\boldsymbol{r}_w$ may not have rank p, and even if they do, some other p columns may be better conditioned. A conceptually elegant algorithm is to compute the QR factorization with column interchanges of $D\,\boldsymbol{r}_w$, where Q is a product of Householder reflections (Rabinowitz 70). The main defect is impossibility of using for sparse matrix calculations.

The descriptions of the algorithms developed by Watson (1986) are presented below for sparse matrixes, and for dense matrices. For small problems (n < 200), it is possible to use the method developed for dense matrices. A big number of practical applications can be handled using dense matrices tools.

Algorithm for dense matrices.

1. Set s:=0,y:=(0,a), ypold:=yp:=(1,0,…,0), restart:=false,error:=initial error tolerance for the ODE solver (.
2. If y_1<0 then move to Step 23.
3. If s>some constant, then
4. S=0
5. Compute a new vector a from ($a=\dfrac{\boldsymbol{l}F(x)+(1-\boldsymbol{l})x}{1-\boldsymbol{l}}$) If
   $\|new\,a-old\ \mathrm{a}\|>1+\text{constant}\ *\|old\ \mathrm{a}\|$, then go to 23
6. ode error:=error.
7. *if* $\|yp-ypold\|_\infty>(last\ arc\ length\ step)*\text{constant}$,*then*
   ode error := *tolerance* << error.

8. ypold:=yp
9. Take a step along the trajectory of (5-7) with ODE solver, yp=y'(s) is computed for the ODE solver by 10-12 .
10. Find a vector z in the kernel of $D\,r_a(y)$ using Householder reflections.
11. If $z^t\, ypold < 0, then$ z := -z.
12. yp:=z/$\|z\|$
13. If the ODE solver returns an error code, then go to Step 23.
14. If $y_1 < 0.99$, then go to 2.
15. If restart=true, then go to 20.
16. restart:=true
17. error:=final accuracy desired.
18. If $y_1 \geq 1$, then set (s,y) back to the previous point (where $y_1 \leq 1$).
19. Go to Step 4.
20. If $y_1 < 1$ then go to Step 2.
21. Obtain the zero (at $y_1 = 1$) by interpolating mesh points used by the ODE solver.
22. Normal return.
23. Error return.

Algorithm for sparse matrix. The general theory applies equally well here. Since the Jacobian matrix has rank n along ã, the derivative $(dx/ds, d\lambda/ds)$ is uniquely determined by Equation 3.73 and continuity, and the initial value problem can be solved for x(s),ë(s). The difficulty now is that the first n columns of the Jacobian matrix $D\,r_a(x, \lambda)$ are definitely special, and any attempt to treat all n+1 columns uniformly would be disastrous from the point of view of storage allocation. Watson used for the $D\,r_a(x, \lambda)$=0 equation solution a preconditioned conjugate gradient method. The description of the method is not presented here because later we will present a detailed review of the linear equations solution methods. The only important detail is the choice of the preconditioning matrix Q. Watson takes for Q the modified Cholesky decomposition of M. For sparse problems the logic of tracking the zero curve ã is exactly the same as for the dense Jacobian matrix.

The normal flow algorithm. As the homotopy parameter vector varies, the corresponding homotopy zero curve ã also varies. This family of zero curves is known as the Davidenko flow (Rabinowitz 1970). The normal-flow algorithm is so called because the iterates converge to the zero curve ã along the flow normal to the Davidenko flow. The normal flow algorithm has four phases: prediction, correction, step size estimation, and computation of the solution at ë=1.

Prediction phase. Assume that several points $P^1 = (\lambda(s_1), Y(s_1)), P^2 = (\lambda(s_2), Y(s_2))$ on í with corresponding tangent vectors (dë/ds,dY/ds) have been found, and h is an estimate of the optimal step to take along í. The prediction of the next point on í is

$$Z^0 = p(s_2 + h) \tag{3.74}$$

where p(s) is the Hermite cubic interpolation $(\lambda(s), Y(s))$ at s_1 and s_2 .

Corrector phase. Starting at the predicted point Z^0, the corrector iteration is

$$Z^{n+1} = Z^n - [D_{rw}(Z^n)]^{\diamond} rw(Z^n), \text{ n} = 0,1,... \quad (3.75)$$

where $[D_{rw}(Z^n)]^{\diamond}$... is the Moore-Penrose pseudoinverse of the p*(p+1) Jacobian matrix D_{r_w}.

Step size estimation phase. A corrector step ΔZ is the unique minimum norm solution of the equation $|D_{rw}|\Delta Z = -rw$. For dense problems the kernel of $|D_{rw}|$ is found by computing a QR factorization and then by back substitution. By applying this QR factorization to ñw and using back substitution again, a particular solution v for Equation 3.75 can be found. When the iteration of Equation 3.75 converges, the final iterate Z^{n+1} is accepted as the next point on í. This approach can not be applied to the sparse matrices.

Finally, the algorithm is following:

1. s:=0, y:=(0,a), h:=0.1, firststep:=true, arcae, arcre:=absolute, relative error tolerances for tracking ã, ansae, ansre:= absolute, relative error tolerances for the answer.
2. If firststep=false then
3. Compute the predicted point Z^0 use initialization procedure else
4. Compute the predicted point Z^0 using a linear predictor based on y=(0,a) and the tangent there.
5. Iterate with Equation 3.75 until either

$$\|\Delta Z^k\| \le arcae + arcre\|Z^k\| \quad \text{or}$$

4 iterations have been performed.

6 If the Newton iteration did not converge in 4 steps, then
7 h:=h/2
8 If h is unreasonably small, then return en error flag.
9 Go to Step 2.
10 firststep:=false.
11 If $y_1 < 1$, then compute a new step size h and go to Step 2.
12 Do Steps 13-18 some fixed number of times
13 Find s such that p(s)=1, using yold, ypold, y, yp in (17)
14 Do two iterations of step (18) starting with $Z^0 = p(s)$, ending with Z^2.
15 If

$$|Z_1^2 - 1| + \|\Delta Z^1\| \le ansae + ansre\|Z^1\|$$

than return (solution has been found)

16 If $Z_1^2 \ge 1$, then
17 Y:= Z^2, *yp* := tangent at Z^2

Else:

18. yold:= Z^2, ypold := tangent at Z^2

19. Return with an error flag.

The logic of the predictor, corrector, and step-size estimation phases of this algorithm is identical to that given in the dense case. Similar to the ordinary differential equation based algorithm, the difference between the dense and sparse Jacobian matrix cases is the low-level numerical linear algebra. The main linear algebra problem is the solution of Equation 3.75 using the same matrix splitting, preconditioning matrix, and conjugate-gradient algorithm as for the sparse ordinary differential equation based algorithm.

Augmented Jacobian matrix algorithm. Augmented Jacobian matrix algorithm has been described by Rheinboldt (1983). This algorithm has four major phases: prediction, correction, step-size estimation, and computation of the solution at ë=1.

Prediction phase. The prediction phase is exactly the same as in the normal flow algorithm.

Correction phase. Starting with the predicted point Z^0, the correction is performed by quasi-Newton iteration defined by

$$Z^{k+1} = Z^k - \begin{bmatrix} A^k \\ T^{(2)t} \end{bmatrix}^{-1} \begin{pmatrix} \rho_a(Z^k) \\ 0 \end{pmatrix}, k = 0,1,... \tag{3.76}$$

where A^k is an approximation to the Jacobian matrix $D\,\rho_a(Z^k)$. The last row of this matrix insures that iterates lie in a hyperplane perpendicular to the tangent vector T^2. Equation 3.76 is the quasi-Newton iteration for solving the augmented nonlinear system

$$\begin{pmatrix} \rho_a(y) \\ T^{(2)t}(y - Z^0) \end{pmatrix} = 0$$

A corrector step ΔZ^k is the unique solution to the equation

$$\begin{bmatrix} A^k \\ T^{(2)k} \end{bmatrix} \Delta Z^k = \begin{pmatrix} -\rho_a(Z^k) \\ 0 \end{pmatrix}$$

Step-size estimation. At each point P^k with tangent T^k along ã, the curvature is estimated by the formula

$$\left\| w^k \right\| = \frac{2}{\Delta s_k} \left| \sin(\alpha_k / 2) \right|$$

where

$$\left\|w^k\right\| = \frac{T^k - T^{k-1}}{\Delta s_k}, \boldsymbol{a}_k = \arccos\left(T^{(k)t} T^{k-1}\right), \Delta s_k = \left\|P^k - P^{k-1}\right\|.$$

This curvature data can be extrapolated to produce a prediction for the curvature for the next step

$$\boldsymbol{x}_k = \left\|w_k\right\| + \frac{\Delta s_k}{\Delta s_k + \Delta s_{k-1}} \left(\left\|w^k\right\| - \left\|w^{k-1}\right\|\right).$$

As with the normal-flow algorithm, additional refinements on the optimal step size are made in order to prevent chattering and unreasonable values.

Computation of the solution at ë=1. The final phase of the algorithm for computation of the solution at ë=1 is entered when a point P^2 is generated such that $P_1^2 \geq 1$. The algorithm for finding this solution is a two-step process that is repeated until the solution is found. First, starting from a point P^k a prediction Z^{k-2} for the solution is generated such that Z^{k-2}=1. Second, a single quasi-Newton iteration is performed to produce a new point P^{k+1} close to ã, but not necessarily on the hyperplane ë=1. The altering process of computing a prediction and taking a quasi-Newton step is repeated until the solution is found.

In summary, the algorithm is:

1. s:=0;; y:=(0,a); ypold:=(1,0); h:=0.1; failed:=false; firststep:=true; arcae, arcre:=absolute, relative error tolerances for tracking ã; ansae, ansre=absolute, relative error tolerances for the answer.
2. Compute the tangent yp at y, and update the augmented Jacobian matrix.
3. If firststep=false then
4. Compute the predicted point
5. Compute the predicted point Z^0 using a linear predictor based on y and yp.
6. If failed=true then
7. Compute the augmented Jacobian matrix at Z^0.
8. Compute the next iterate Z^1
9. Limit:=2([- $\log_{10}(arcae + arcre\left\|y\right\|)$]+1). Repeat steps 10-11 until either
   $$\left\|\Delta Z^k\right\| \leq arcae + arcre\left\|Z^k\right\|$$
   or
   limit iterations have been performed
10. Update the augmented Jacobian matrix.
11. Compute the next iteration
12. If the quasi-Newton iteration did not converge in limit steps, then
13. h:=h/2; failed:=true.
14. If h is unreasonably small, then return with an error flag.
15. Go to Step 3.
16. Compute the tangent at the accepted iterate Z^* and update the augmented Jacobian matrix
17. Compute the angle á between the current and previous tangents

18. If á>ð/3, then
19. H:=h/2; failed:=true.
20. If h is unreasonably small, then return with an error flag.
21. Go to Step 3.
22. Yold:=y, ypold:=yp, y:= Z^*, yp:=tangent computed in step 16, firststep:=false, failed:=false.
23. If $y_1 < 1,$ then compute a new step size h with

$$\varepsilon_{\min} = 0.01,$$

$$\delta_k = \min\left\{\left(arcae + arcre\|y\|\right)^{1/4}, \frac{1}{2}\|y - yold\|\right\},$$

and go to Step 3.

24. Find s such that p(s)=1, using yold, ypold, y and yp yopp:=yold, Z^0 =p(s).
25. Limit:=2([- $\log_{10}(arcae + arcre\|y\|)$]+1). Do steps 26-33 for k=2,…,limit+2.
26. Update the augmented Jacobian matrix
27. Take a quasi-Newton step.
28. If

$$\left|P_1^{k+1} - 1\right| + \left\|\Delta Z^{k-2}\right\| \le ansae + ansre\left\|Z^{k-2}\right\|$$

then return (solution has been found)

29. If $\left|P_1^{k+1} - 1\right| \le ansae + ansre,$

Then

$$Z^{k-1} := P^{k+1}$$

else do steps 30-33.

30. yold:=y, y:= P^{k+1}
31. If $yold_1$ and y_1 bracket $\lambda = 1,$ then yopp:=yold.
32. Compute Z^{k-1} with the linear predictor using y and yold.
33. If $\left\|Z^{k-1} - y\right\| > \|y - yopp\|,$ then compute $Z^{(k-1)}$ with the linear predictor using y and yopp
34. Return with an error flag.

Augmented sparse Jacobian matrix algorithm. The augmented Jacobian matrix algorithm for sparse Jacobian matrices differs from the dense algorithm in the following three respects:

1. Like the sparse normal flow and ODE-based algorithms, the low-level numerical linear algebra is changed to take advantage of the sparsity of the problem.
2. Quasi-Newton iterations are abandoned in favor of pure Newton iterations;
3. Rheinboldt's step size control is implemented more faithfully because of the use of Newton iterations.

Except for these three changes, the logic for tracking the zero curve ã is exactly the same as for the dense algorithm.

Conclusions. The homotopy algorithm the most robust between the other algorithms. The most useful and effective the algorithm based on the augmented Jacobian, especially with the new methods for augmented matrices handling, developed in the last years.

3.2.8 Solving linear equations

All the methods for solving nonlinear equations involve the solution of many sets of linear algebraic equations. When the number of equations (n) becomes very large, it is clear that two difficulties can occur: the computation time required may become too long and/or there may be not enough storage to contain the matrices and factorizations required. Methods for the solution of the linear equations

$$Ax = b \tag{3.77}$$

can be divided into two categories: direct methods where matrix A is transformed into a matrix that is easily invertible, and iterative methods where matrix A is used directly to generate a sequence of vectors which tend to the solution of equations.

Another type of methods is based on the division of the matrices by symmetric and asymmetric positive definite and indefinite. The most effective methods in each of these categories will be described where A is a large sparse matrix, which is usual situation for network problems.

Direct methods. It is important to distinguish between two types of matrices:
A stored matrix is one for which all n^2 matrix elements a_{ij} are stored in the computer memory. It is very important to use standard building blocks in application codes. They are extremely useful for simplifying the design of codes while guaranteeing portability and efficiency. The building blocks for much of our work, both in the solution of sparse systems, and in more complicated are as of constructing and solution the network problems, are the Basic Linear Algebra Subprograms known as BLAS (Dongarra 1990). For reasons of efficiency, we are going to use the higher level of BLAS, in particular, the LEVEL 3 BLAS that includes kernels like the matrix-matrix multiply routine. Later we will discuss the effect of using BLAS in the solution of linear equations when the coefficient matrix is sparse.

A sparse matrix is one for which most of the matrix elements are zero, and nonzero elements can be either stored in some special data structure or regenerated as needed. The order of n is frequently as large as several tens of thousands and occasionally even larger. This type of matrices the more important for network problems and will be discussed below. The significant benefit from sparsity does not come from the cost reductions, but rather from the fact that problems that were hitherto infeasible can now be solved. For network problems there was a near-linear relationship between the number of floating-point operations and the number of unknowns in the sparse system of equations. Matrix sparsity and graph theory are subjects that can be closely linked. The pattern of a square sparse matrix can be represented by a graph, for example, and then the results from graph theory can be used to obtain sparse matrix results. Here we will not discuss the computer architecture and its influence on the algorithms. We will be restricted to the personal computers and will not touch the problems arising from the mainframe computer architecture.

Sparsity is useful not only in solving the equations efficiently, but also in reducing the number of function evaluations necessary to estimate Jacobian matrix of the nonlinear system of equations. An algorithm due to Curtis, Powel can accomplish this, and Reid who is discussed later (Rabinowitz 1970).

According to Duff: “I feel strongly that the only way of solving really challenging linear algebra problems

is by combining direct and iterative methods through either conventional or novel preconditioning"(Duff 1996).

Unifrontal methods. In a unifrontal scheme the factorization proceeds as a sequence of partial factorization and elimination on dense submatrices, called frontal matrices. Although unifrontal methods were originally designed for the solution of finite-element problems, they can be used on assembled systems. For assembled systems, the frontal matrices can be written as

$$\begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix} \tag{3.78}$$

where all rows are fully summed (that is there are no further contributions to come to the rows in Matrix 3.78 and the first block column is fully summed. This means that pivots can be chosen from anywhere in the first block column and, within these columns, numerical pivoting with arbitrary row interchanges can be accommodated since all rows in the frontal matrix are fully summed. Let's assume that without loss of generality the pivots, chosen in the square matrix F_{11} of (3.78) F_{11}, are factorizes, and the Gaussian elimination multipliers overwrite F_{21} and the Schur complement

$$F_{22} - F_{21}F_{11}^{-1}F_{12} \tag{3.79}$$

is formed using dense matrix kernels. The submatrix consisting of the rows and columns of the frontal matrix from which pivots have not yet been selected is called the contribution block. In the case above, this is the same as the Schur complement matrix.

At the next stage, further rows from the original matrix are assembled with the Schur complement to form another frontal matrix. The frontal matrix is extended in size, if necessary, to accommodate the incoming rows. The overhead is low since each row assembled only once and there is never any assembly of two (or more) frontal matrices. The entire sequence of frontal matrices is held in the same working array. Data movement is limited to assembling rows of the original matrix into frontal matrix, and storing rows and columns as they become pivotal. There is never any need to move or assemble the Schur complement into another working array. One important advantage of the method is that only this single working array need reside in memory. Rows of A can be read sequentially from disk into the frontal matrix. Entries in L and U during LU decomposition of A can be written sequentially to disk in the order they are computed.

The unifrontal method works well for matrices with small profile, where the profile of a matrix is a measure of how close the nonzero entries are to the diagonal and is given by the expression:

$$\sum_{i=1}^{n} \left\{ \max_{a_{ij} \neq 0} (i-j) + \max_{a_{ji} \neq 0} (i-j) \right\} \tag{3.80}$$

where it is assumed the diagonal is nonzero so all terms in the summation are non-negative. For matrices that are symmetric or nearly so, the unifrontal method is typically preceded by an ordering method to reduce the profile such as reverse Cuthill-McKee (RCM) (George 1973). This is typically faster than the

sparsity-preserving orderings commonly used by a multi-frontal method (such as nested dissection and minimum degree). However, for matrices with large profile, the frontal matrix can be large, and an unacceptable amount of fill-in occurs. In particular, we lack effective profile reduction strategies for matrices whose pattern is very asymmetric.

The unifrontal scheme can easily accommodate numerical pivoting. Because all rows in Matrix 3.78 are fully summed and regular pivoting can be performed, it is always possible to choose pivots from all the fully summed columns.

In a multifrontal method, the sparse factorization proceeds by a sequence of factorizations on small full matrices, called frontal matrices. An important feature of these methods is that arithmetic is performed on dense submatrices and Level 2 and Level 3 BLAS (matrix-vector and matrix-matrix kernels) can be used. Both sparse LU and QR factorizations can be implemented within this framework. The ordering for the sequence of computations and the frontal matrices are determined by a computational tree, called assembly tree, where each node represents a full matrix factorization and each edge transfers data from child to parent node. This assembly tree is determined from the sparsity pattern of the matrix and from a reordering that reduces the fill-in during the numerical factorization (such as the minimum degree ordering). During the numerical factorization, elimination at any node can proceed as soon as those at the child nodes have completed and the resulting contributions from the children have been summed (assembled) with the data at the parent node. This is the only synchronization that is required and means that operations at nodes that are not ancestors or dependents are completely independent.
The development and production of sparse asymmetric solvers has now become a veritable growth industry. The multi-frontal and super-nodal techniques, so powerful in the symmetric case, have been extended to asymmetric systems (Duff 1996).

The solution of a sparse matrix system is usually divided into several phases:
Analysis of the sparsity structure to determine a pivot ordering.
Symbolic factorization to generate a structure for the factors.
Numerical factorization.
Solution of set(s) of equations.

In some cases, particularly when it is important to consider numerical values when choosing pivots, the first three phases are combined into an analyze-factorize phase. Additionally, there may be some partitioning scheme prior to all these phases, which are then executed on submatrices from the partition. The relative speeds of the four phases are very dependent on the details of the algorithm and implementation, the problem being solved, and the machine being used. However, one reason for separating the first two phases from the third is that it is usually much faster to perform an analysis and symbolic factorization without reference to numerical values. This does mean that there must be some way of incorporating subsequent numerical pivoting in the factorization if general problems are to be solved. Most of the algorithms are based on matrix factorization methods like Gaussian elimination, and part-way through the matrix factorization, when some of the factors are calculated, the remaining matrix is composed of original matrix entries and filled-in from the earlier stages. Let us refer to this matrix as reduced matrix.

Finally, the relationship between graphs and sparse matrices is ubiquitous in sparse matrix work. Using a symmetric sparse matrix of order n, we associate a graph with n vertices, and an edge (I,j) between

vertices I and j, if and only if entry $a_{ij} \neq 0$. A clique is a complete subgraph, that is all vertices of the subgraph are pairwise connected by edges in the graph. Asymmetric matrix we associate with directed graph where the edges are now directed. A bipartite graph is sometimes associated with an asymmetric (even non-square) matrix. This has two sets of disjoint vertices identified with rows and columns of the matrix respectively. An edge (I,j) exist from the row vertices to the column vertices if and only if entry $a_{ij} \neq 0$. The elimination tree is defined by the Cholesky (George 1973) factors of a symmetric matrix and has an edge (I,j) if the first nonzero entry below the diagonal in column j of the lower triangular Cholesky factor is in row i.

Three principal problems with multi-frontal methods, as practiced in the 1980s, were that
there could be significant overheads for data movement,
they assumed structural symmetry, and
the analyzes phase assumed that any diagonal entry could be chosen as pivot.

In the first case, continuing with one frontal matrix rather than stacking it and starting another can reduce data movement. Taken to the extreme, this strategy would give a uni-frontal scheme. This technique was incorporated in the HSL code MA38. This is clearly fine if the matrix is symmetric in structure but surprisingly can perform quite well for asymmetric matrices. Although it can be inefficient when the matrix is markedly asymmetric. For asymmetric systems, a preordering to place non-zeros on the diagonal is often very helpful and is in option in the HSL code MA41. Davis and Duff (1993) have extended the multifrontal scheme to general asymmetric matrices, using rectangular fronts and directed cyclic graphs. This extension works well on most highly asymmetric systems but can be poorer then MA41 when the matrix is not highly asymmetric.

The third problem is more difficult and is present for any technique that performs an analyze phase separation from the numerical factorization. Clearly, one way to resolve this is to combine the analyze and factorize phases but then many of the benefits of the fast analyze phase are lost. A certain amount of numerical or additional structural information can be supplied to the analyze phase.

The other main approach to using higher order level BLAS in sparse direct solvers is generalization of sparse column factorization. These can either be left-looking (or fan-in) algorithms, where updates are performed on each column in turn by all the previous columns that contribute to it. Then the pivot is chosen in that column and the multipliers calculated; or a right-looking (or fan-out) algorithm where, as soon as the pivot is selected and multipliers calculated, that column is immediately used to update al future columns that is modifies. Higher level BLAS can be used if columns with a common sparsity pattern are considered together as a single block or super node and algorithms are termed column-super node, super node-column, and super node-super node depending on whether target, source, or both are super nodes.

One of approaches for handling duct nets is the use of symmetrical matrices. Such symmetrical matrix is presented below.

Ordering for symmetric problems. Although predated thirty years ago (Markowitz 1957), scheme S2 in the paper by Tinney and Walker (1967) established the main ordering for symmetric problems that has remained almost unchallenged until this present day. Scheme S2 is commonly termed the minimum degree ordering because, at each stage, the pivot chosen corresponds to a node of minimum degree in the

undirected graph associated with the reduced matrix. In matrix terms, this corresponds to choosing the entry from the diagonal that has the least number of entries in rows within the reduced matrix. This ordering algorithm has proved remarkably resistance and, although based only on local criterion, does an excellent job of keeping subsequent work and fill-in over a wide range of problems. George and Liu (1989) study the evolution of the minimum degree ordering.

George (1973), in his later paper, proposed a different class of ordering based on a non-local strategy of dissection. In his nested dissection approach, a set of nodes is selected to partition the graph, and this set is placed at the end of the pivotal sequence. The subgraphs corresponding to the partitions are themselves similarly partitioned and this process is nested with pivots being identified in reverse order. Minimum degree, nested dissection, and several other symmetric orderings were included in the SPARSPAK package. Empirical experience at the beginning of the 90s indicated that minimum degree was the best ordering method for general or unstructured problems which is the class of problems under our application.

Although the minimum degree ordering is simple enough to describe, it is not quite so simple to implement efficiently. There are three main issues:
select a pivot,
update a reduced matrix after selection of a pivot, and
update the degree counts.

For the first, it is easy to keep a list of nodes in order of increasing degree and to choose the node at the head of that list each time. There are two ways in which this task can be made significantly more efficient. The first is to observe that, once a node is selected, all nodes that were in a complete subgraph (or clique) containing this node, have one degree less and so can immediately be eliminated without any extra fill-in, and subsequently, all the nodes in the clique can be eliminated. This is usually termed mass node elimination and was included in some early minimum degree codes. It is quite natural, in a finite element like approach, to perform the minimum degree ordering on a graph where nodes in the same degree clique are treated as a single node, and this was done by the minimum degree ordering algorithm in the HSL code MA47 (Duff 1995). The second improvement to pivot selection stems from the observation that, if two nodes of the same degree are not adjacent in the graph, they can be eliminated simultaneously. This is termed multiple elimination.

The resolution of the second issue, graph update, was the main reason why minimum degree codes improved by several orders of magnitude over the decade 1976-1986. The principal saving was made by using the clique structure of the reduced matrix and updating this rather than individual edges of the graph. This was discussed in the review by Reid (1987).

There are two main approaches to updating the degree counts. One is too have a threshold and compute the new degrees only if they could fall below this threshold. The threshold must, of course, be changed dynamically, at which point some recalculation of degrees is necessary. The second is to replace the minimum degree count by an approximate degree count that is easier to compute. Recently Amestoy, Davis, and Duff (Davis 1995) have designed an approximate minimum degree ordering (AMD) where the bound is equal to the degree in many cases. They have found that their AMD ordering is almost indistinguishable from the minimum degree ordering in equality but it is much faster to compute. It is only within the last year that the supremacy of minimum degree has been challenged. The beauty of dissection orderings is that they take a global view of the problem; their difficulty until recently has been the problem of extending them to unstructured problems. Recently, there have been several tools and approaches that

make this extension more realistic.

The essence of a dissection technique is a bisection algorithm that divides the graph of the matrix into two partitions. If node separators are used, a third set will correspond to the node separators. Such a bisection is then repeated in nested fashion to obtain an ordering for the matrix. Perhaps the bisection technique that has achieved the most fame has been spectral bisection. In this approach, use is made of the Laplacian matrix that is defined as a symmetric matrix whose diagonal entries are the degrees of the nodes and whose off-diagonals are -1 if and only if the corresponding entry in the matrix is nonzero. This matrix is singular because its row sums are all zero, but if the matrix is irreducible, it is positive semi-definite with only one zero eigenvalue (called the Fiedler vector). This matrix can be used to define a bisection by constructing a vector x that has components x_i equal to +1 or -1, according to which partition node I lies in. Then the quantity $x^T Ax$ is 4 times the number of edges between the two halves of the bisection. We can thus obtain an optimal bisection by minimizing $x^T Ax$ subject to $\sum_i x_i = 0$ with $x_i = \pm 1$. Since the first constraint corresponds to finding a vector orthogonal to the vector of all ones, which is the eigenvector for the zero eigenvalue, in the corresponding continuous problem it is the eigenvector corresponding to the smallest nonzero eigenvalue that is of interest. Normally some variant of Lanczos algorithm is used to compute this (Duff 1996). The graph is then bisected according to the components of this eigenvector.

The spectral method requires much computing time, does not always yield optimal bisections, and naturally produces edge separators, requiring some postprocessing to obtain a node separator set. For these reasons, this technique is not strongly favored, and there has been much current research on alternatives. But for the duct net task it can be approved. The main reason of this is the necessity to make many calculations for the duct net having topology fixed. This seems reasonable, to spend a comparatively big time for preprocessing, but after this have a benefits of better ordering for a big number of different runs.

The other main approach to graph bisection is to perform graph reductions, compute a partition cheaply on the resulting coarse graph, and from this construct a partition of the original graph, using some kind of iterative improvement on the projection of this coarse partition on the finer graph. This approach is nested and is termed a multilevel scheme. Multilevel schemes are very interesting and promising. Later we will discuss them in application to the Newton method.

Solution of sets of sparse asymmetric equations. In the case of asymmetric systems usually a Markowitz ordering (Markowitz 1957) or a column ordering based on minimum degree on the normal equations is used together with a threshold criterion for numerical pivoting. The main recent activities for asymmetric systems have been the development of methods based on partitioning and the production of efficient codes using higher level BLAS operations. A decade ago, the only widespread use of partitioning was to preorder the matrix to block triangular form prior to perform the analyze-factorize phase on the block on the diagonal of that form. A major problem with this approach is that the partitioning does not guarantee that the diagonal blocks are well conditioned, or even nonsingular. Thus some a posterior measures must be taken to improve the stability of the factorization. This issue of stability was discussed by Arioli, Duff, Gould and Reid (1990) and, more recently, methods for maintaining stability have been developed by Hansen, Ostromsky and Zlatev (1994).

A related approach is to expand the matrix, perhaps artificially, in order to obtain a system that is larger

and sparser. Normally, there is a choice of pivots for this system that would reduce it to the original system. The logic of this that we might be able to do better by using the extra degree of freedom on the expanded system. A simple example of this matrix stretching technique is the augmented system.

A major tool in the efficient implementation of codes for asymmetric systems has been the observation presented by Gilbert and Pejerls (1988) that partial pivoting can be performed in time proportional to the number of arithmetic operations, so avoiding any potentially costly sorting operations. A refinement of this technique was provided by Eisenstat and Liu (1992), who suggested ways of pruning a search tree to reduce work in the symbolic phase. Variants of this technique are used in nearly all-sparse partial pivoting codes.

Frontal, multifrontal, and supernodal approaches for the solution of asymmetric problems have all seen significant recent development. The HSL frontal code MA42, which solves asymmetric systems, was redesigned to use standard Level 2 and Level 3 BLAS and can accommodate entry by both equations and elements. If the matrix is close to symmetric in structure in the sense that a_{ij} is usually nonzero then a_{ji} is, then methods adapted from symmetric multi-frontal approaches can be used, with the analyze phase performed on he pattern of $A+A^T$. This approach can work well for the duct net case (almost symmetric).

Preconditioning. One of the main problems with sparse LU factorization is that often the number of entries in the factors is substantially greater than in the original matrix so that, even if the original matrix can be stored, the factors cannot. If the original matrix can be stored, the possibility is to start a sparse LU factorization but drop some fill-in entries so that the partial factors can still be stored. Algorithms for doing this are called incomplete LU (or ILU) factorization and they differ depending on the criteria for deciding which entries to drop. At one extreme, we could hold all the factors, while at the other we could store no fill-ins. This partial or incomplete factorization is then used to precondition the matrix for iterative solution, normally using a fairly standard Krylov-sequence based iterative technique like conjugate gradients in the symmetric case or GMRES or BiCG when the matrix is asymmetric.

The main criteria for deciding which entries to include in an incomplete factorization are location and numerical value. The commonest location-based criterion is to allow a set number of levels of fill-in, where original entries have level zero, original zeros have level and a fill-in position (I,j) has level $Level_{ij}$ determined by

$$\min_{1\le k\le \min(i,\,j)} \left\{ Level_{ik} + Level_{kj} + 1 \right\} \tag{3.81}$$

The other main criterion for deciding which entries to omit is to drop entries less than a predetermined numerical value. Since it is usually not known a priori how many entries will be above a selected threshold, the dropping strategy is normally combined with restricting the number of fill-ins allowed to any one column. When using a threshold criterion, it is possible to change it dynamically during the factorization to attempt to achieve a target density of the factors. Although the notation is not yet fully standardized, the nomenclature commonly adopted for incomplete factorization is ILU(k). It happened when k levels of fill-in are allowed and ILUT(á,f), for the threshold criterion when entries of modulus less than á are dropped and the maximum number of fill-ins allowed in any columns is f.

The use of incomplete factorizations as pre-conditioners for symmetric systems has a long pedigree and good results have been obtained for a wide range of problems. An incomplete Cholesky factorization where one level of fill-in is allowed (ICCG(1)) has proven to provide a good balance between reducing the number of iterations and the cost of the computing and using the preconditioning. The situation with symmetric systems is quite well analyzed and understood.

The situation for asymmetric systems are, however, much less clears. There have been many experiments on using incomplete factorizations and there have been studies of the effect of orderings on the number of iterations that show similar behavior to the symmetric case. However, there is very little theory governing the behavior for general systems and indeed the performance of ILU pre-conditioners is very unpredictable. In fact, a mayor problem is that the ILU factors can be more ill-conditioned than the original system and so the preconditioned system can perform much worse than the original matrix with respect to convergence of the iteration method.

The QR or LQ factorizations can also be used in the asymmetric case to derive an incomplete factorization. Here the orthogonal factor need not be kept and the resulting incomplete triangular factor can be used to precondition the normal equations. Pre-conditioners based on those factorizations are generally more expensive to compute but they are usually more robust. One way of computing an incomplete orthogonal factorization is to use incomplete modified Gram-Shmidt (IMGS) (Ostrowski 1966).

Another whole class of pre-conditioners are those where the direct method is used to solve a sub-problem of the original problem. This is often used in a domain decomposition setting, where problems on sub-domains are solved by the direct method but the interaction between the sub-problems is handled by an iterative technique. A related example of this is the block projection methods.

Multi-grid techniques also often combine aspects off both iterative and direct methods. These methods were originally developed for solving partial differential equations but developments such as algebraic multi-grid extend their applicability to more general systems. The basic idea is to use corrections on a sequence of coarser grids to update the required solution on a fine grid. It is common to use a direct method for the solution on the coarsest grid with one or two iterations of usually a simple iterative method on the other grids.

Iterative methods. Iterative methods can be divided for stationary properties not changed with iterations and nonstationary matrix of the transformation changed after every some iterations methods and non-stationary methods.

Stationary methods

Jacobi. The Jacobi method is based on solving for every variable locally with respect to the other variables; one iteration of the method corresponds to solving for every variable once. The resulting method is easy to implement but convergence is slow.

Gauss-Seidel. The Gauss-Seidel method is like Jacobi method, except that it uses updated values as soon as they are available. In general the Gauss-Seidel method will converge faster than Jacobi, though still relatively slowly.

SOR. Successive over-relaxation (SOR) can be derived from the Gauss-Seidel method by introducing an extrapolation parameter ù. For the optimal choice of ù, SOR may converge faster than Gauss-Seidel by an order of magnitude.

SSOR. Symmetric successive over-relaxation (SSOR) has no advantage over SOR as a stand alone iterative method; however, it is useful as a pre-conditioner for non-stationary methods.

The above mentioned methods are comparatively old. There are newer much more powerful and effective tools based on non-stationary methods.

Non-stationary methods.
Non-stationary methods differ from stationary methods in a way they handle the information from the changes at each iteration. Typically, constants are computed by taking inner products of residuals or other methods arising from the iterative method. Description below is presented from Dongarra (1995).

Conjugate Gradient (CG). The linear conjugate gradient method was proposed by Hestenes and Stiefel in the 1950s and is considered as the fundamental importance in scientific computing. Its performance is strongly tied to the choice of the so-called pre-conditioner, which determines the asymptotic rate of the convergence. The first nonlinear conjugate gradient method was introduced by Fletcher and Reeves in 1960s, and is one of the earliest techniques for solving large nonlinear optimization problems. Several variations of the original method have been proposed and are widely used in practice. The importance of the non-linear conjugate gradient methods lies in the fact that they require no matrix storage and are faster than the steepest descent method.

The Conjugate Gradient method proceeds by generating vector sequences of iterates, residuals corresponding to iterates, and search directions used in updating the iterates and residuals. Although the length of these sequences can become large, only a small number of vectors need to be kept in memory. In every iteration of the method, two inner products are performed in order to compute update scalars that are defined to make the sequences satisfy orthogonal conditions. The pseudo-code for the Preconditioned Conjugate Gradient Method is follow:

Compute $r^{(0)} = b - A x^{0}$ for some initial guess x^{0}

For I=1,2,…

Solve $M x^{(i-1)} = r^{(i-1)}$

$\boldsymbol{r}_{(i-1)} = r^{(i-1)^{T}} x^{(i-1)}$

if I=1

$\boldsymbol{r}^{(1)} = x^{(0)}$

else

$\boldsymbol{b}_{i-1} = \boldsymbol{r}_{i-1} / \boldsymbol{r}_{i-2}$

$\boldsymbol{r}_{i} = z^{(i-1)} + \boldsymbol{b}_{(i-1)} \boldsymbol{r}_{(i-1)}$

endif

$q^{(i)} = A p_{(i)}$

$$\alpha_i = p_{(i-1)} / p_i^T q^i$$
$$x^i = x^{i-1} + \alpha_i p^i$$
$$r^i = r^{i-1} - \alpha_i q^i$$

check convergence; continue if necessary.

For the Conjugate Gradient method, the error can be bounded in terms of the spectral condition number è of the matrix $M^{-1}A$. If ë1 and ë0 are the largest and smallest eigenvalues of a symmetric positive definite matrix B, then the spectral condition number of B is è=ë1/ë0

Minimum Residual (MINRES) and Symmetric LQ (SYMMLQ). When A is not positive definite, but symmetric, one can still construct an orthogonal basis for the Krylov subspace by three term's recurrence relations. In matrix form recurrence can be written as

$$A R_i = R_{i+1} T_i$$

where T_I is an (I+1)⊗I tri-diagonal matrix is an tri-diagonal matrix. But a problem exists that an inner product no longer defines. However, we can still try to minimize the residuals in the 2-norm by obtaining

$$x^i \in \{r^0, A r^0, ..., A^{i-1} r^0\}, x^i = R_i y$$

that minimizes

$$\|A x^i - b\|_2 = \|R_{i+1} T_i y - b\|_2$$

Here is the fact that, if $D_{i+1} \equiv diag(\|r^0\|_2, \|r^1\|_2, ..., \|r^i\|_2)$, then $R_{i+1} D_{i+1}^{-1}$ is an ortho-normal transformation with respect to the current Krylov subspace exploited as: the element in the (I+1,I) position of T_i can be annihilated by a simple Givens rotation and the resulting upper bi-diagonal system (the other sub-diagonal elements having been removed in previous iteration steps) can simply be solved, which leads to the MINRES method.

Another possibility is to solve the system $T_i y = \|r^0\| e^1$, as in the CG method (T_i is the upper i*i part of $\overline{T}_i$). Other than in CG we cannot rely on the existence of a Cholesky decomposition (since A is not positive definite). An alternative is to decompose T_i by an LQ-decomposition. This again leads to simple recurrences and the resulting method is known as SYMMLQ .

Conjugate Gradient on the Normal Equations: CGNE and CGNR. If a system of linear equations Ax=b has a non-symmetric, possibly indefinite (but nonsingular) coefficient matrix, one obvious attempt at a solution is to apply Conjugate Gradient to a related symmetric positive definite system, $^TAx=A^Tb$. While this approach is easy to understand and code, the convergence speed of the Conjugate Gradient method

now depends on the square of the condition number of the original coefficient matrix. Thus the rate of convergence of the CG procedure on the normal equations may be slow.

Several proposals have been made to improve the numerical stability of this method. The best known is based upon applying the Lanczos method to the auxiliary system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

A clever execution of this scheme delivers the factors L and U of the LU-decomposition of the tri-diagonal matrix that would have been computed by carrying out the Lanczos procedure with A^TA Another means for improving the numerical stability of this normal equations approach is based on the observation that the A^T A matrix is used in the construction of the iteration coefficients through an inner product like (p,A^TAp). This leads to the suggestion that such an inner product be replaced by (Ap,Ap).

Below a short review the common approach will be presented. Using this approach almost all modern numerical methods for iterative solution of the linear equations can be developed.

Krylov subspace methods. Krylov subspace based methods can be viewed as polynomial-based iterative schemes for solving systems of the form Ax=b, of dimension n. The general form of a polynomial iterative scheme is given by

$$x_k = x_{k-1} + \sum \beta_{kj} r_j$$

(3.82)
where $r_j = b - A x_j$

Krylov subspace based methods compute new approximations to the solution, x_k, from the affine subspace defined by

$$x_0 + K_k(r_0, A) \tag{3.83}$$

where the Krylov subspace of dimension k is defined by

$$K_k(r_0, A) \equiv span(r_0, A r_0, A^2 r_0, \ldots, A^{k-1} r_0), \tag{3.84}$$

and r_0 is the initial residual determined from the initial solution guess, x_0.

Two different approaches taken in the derivation of Krylov algorithms; namely, the minimal residual approach and the orthogonal residual approach. The most promising are the algorithms based upon the Arnoldi process (GMRES) and the non-symmetric Lanczos process (CGS, Bi-CGSTAB, TFQMP).
The most well known methods from this family: Method of steepest descent, Gauss-Seydel-type iteration, Conjugate direction methods, and Orthodir

Conclusions:

1. The universal tool for linear system of equations solution is not exist.
1. Choice for linear equations solver problem dependent and must be performed according the task parameters by the code itself.
2. Parameters significant for the choice of the method are
   - number of equations
   - sparsity level
   - non-linearity of the main task level
   - warm start possibility
   - number of nonzero members in the main matrix and in the factorization
3. For linear equations solution the next methods can be used
   - for the unknowns number of several thousands is the combined unifrontal/multifrontal algorithm
   - for the unknowns number of n*10,000 is the Krylov's subspace methods with preconditioning based on incomplete factorizations
   - for the unknowns number of n*100,000 is the Krylov's subspace methods with preconditioning based on multi-grid or other many level approach.
5. The BLAS must be used as a basis for all condense matrixes calculations.
6. Sparsity can be effective used not only for equations solution but for Jacobian calculation as well.

The method for solution of linear systems must be task dependent. The different tasks approximately can be classified as

- small tasks (number of unknowns is $\leq$ 1000)
- medium tasks (number of unknowns $\leq$ 10,000)
- big tasks (number of unknowns $\leq$ 100,000)
- super big (number of unknowns > 100,000)

For small tasks it is possible to use simple methods, based on full mode standard software from LINPACK, based on direct solution techniques. Whenever the Jacobian shows no special structure but turns out to be sparse and the size of the task medium or big sparse mode elimination techniques may be successfully applied. As example, it is possible to use well-known MA28 package developed by Duff. In the symmetrical case task the MA27 may be used. For the big and especially super big tasks the Krylov's method must be applied.

3.3 Control systems

Problem Formulation and Derivation of the Necessary Condition. For duct net design control task it is possible to write the following

$$\begin{aligned} &\text{Min } F(y,u) \\ &C(y,u)=0 \\ &y> 0,\ u > 0 \end{aligned} \tag{3.85}$$

where:

$y \in R^m$

$u \in R^{n-m}$

n and m are positive integers satisfying n > m.

The functions f and C are considered smooth and defined as $f : \Omega \to R$, and $C : \Omega \to R^m$, where Ω is an open set of R^n containing $\{y : y \geq 0\} \times \{u : u \geq 0\}$.

For the duct net case:
y is the vector of state variables,
u is the vector of control variables,
C(y,u)=0 is the discreetized state equation,
F(y,u) the sum of squares of the defects of equations $y_I=q_i$.

If the solution of Equations 3.85 have F(x,u) > 0, control is failed to handle control variables in the limited needed. Problem described by Equations 3.85 can be solved by interior point Newton algorithm according to the Vicente (1996).

Constraint qualifications and optimality conditions. In the notation there is

$$x = \begin{pmatrix} y \\ u \end{pmatrix} \tag{3.86}$$

Also, $(z)_y$ and $(z)_w$ represent the sub-vectors of $z \in R^n$ corresponding to the y and u components, respectively, and I_p represents the identity matrix of order p. The Lagrangian of f(x) with respect to the equality constraints C(x)=0 is represented by $l(x,ë)=f(x)+ë^T C(x)$. The Jacobian matrix of C(x) is denoted by J(x). Due to the partition of x in y and u, following takes place

$$J(x)=(C_y(x)\ C_u(x)) \tag{3.87}$$

Assumptions, applied to this approach.
Assumptions 1.

1. The functions f and C are twice continuously differentiable with Lipschitz second derivatives in Ω.
2. The partial Jacobian $C_y(x)$ is nonsingular in Ω.

The method description. Let U be an open set containing {u: u>0} such that for all $u \in U$ exist a solution y of C(y,u)=0 and such that the matrix $C_y(x)$ is invertible for all $x=\left(y^T, x^T\right)^T$ with $u \in U$ and C(y,u)=0. Then exist a twice continuously differentiable function

$$y : U \to R^n \tag{3.88}$$

defined by

C(y(u),u)=0.

This allows reducing the minimization problem to the space of the control variables u

Minimize f(y(u),u)
Subject to y(u)>=0,u>=0. (3.89)

The optimality conditions and constraint qualifications used by Vicente (1996)) are Karush's, Kuhn's, and Tucker's (KKT) conditions. .
The diagonal matrix $D(x, \lambda)$ with diagonal elements is described by

$$(D(x,\lambda))_{ii} = \begin{cases} (x_i)^{\frac{1}{2}} & \text{if } (\nabla_x l(x,\lambda))_i \geq 0 \\ 1 & \text{if } (\nabla_x l(x,\lambda))_i \leq 0 \end{cases}$$

(3.90)

Another significant diagonal matrix is $E(x, \lambda)$ of order n with diagonal elements described by

$$(E(x,\lambda))_{ii} = \begin{cases} (\nabla_x l(x,\lambda))_i \text{ if } (\nabla_x l(x,\lambda))_i > 0 \\ 0 \text{ otherwise,} \end{cases}$$

(3.91)

Proposition 1. A non-degenerate point x_*, with corresponding multipliers λ_*, satisfies the second-order sufficient optimality conditions if and only if it satisfies the first-order conditions and

$$D(x_*, \lambda_*)\nabla^2_{xx} l(x_*, \lambda_*)D(x_*, \lambda_*) + E(x_*, \lambda_*)$$

(3.92)

is a positive definite on the null space of $J(x_*)D(x_*, \lambda_*)$.

Proposition 2. If x_* is a regular point, then the matrix $D(x_*, \lambda_*)J(x_*)^T$ has full column rank.
If the regular point x_*, with corresponding Lagrange multipliers λ_*, is such that Matrix 3.92 is positive definite, then the matrix

$$\begin{pmatrix} D(x_*,\lambda_*)^2 \nabla^2_{xx} l(x_*,\lambda_*) + E(x_*,\lambda_*) & D(x_*,\lambda_*)\ J(x_*)^T \\ J(x_*)D(x_*,\lambda_*) & 0 \end{pmatrix}$$

is nonsingular.

The Newton's method applied to the system of nonlinear equations in x and ë

$$D(x, l)^2 \nabla_x l(x, l) = 0$$

(3.93)

C(x)=0

(3.94)

This algorithm is of the interior-point type, meaning that x is required always to be strictly feasible with respect to the bound constraints. A linearization of Equations 3.93 and 3.94 is of the form

$$\begin{pmatrix} D(x, l)^2 \nabla_{xx}^2 l(x, l) + E(x, l) & D(x, l)^2 J(x)^T \\ J(x) & 0 \end{pmatrix} \begin{pmatrix} s \\ \Delta l \end{pmatrix} = -\begin{pmatrix} D(x, l)^2 \nabla_x l(x, l) \\ C(x) \end{pmatrix} \quad (3.95)$$

the second linear equations in Expression 3.95 is the linearized state equation. The first linear equation in Expression 3.95 has been derived by applying the product rules to Equation 3.93 the case where differentiability is existing.

Let us use the next notation: $\bar{s} = D(x, l)^{-1} s$

After simple transformations it is possible to have from Expression 3.95:

$$\begin{pmatrix} D(x, l)\ \nabla_{xx}^2 l(x, l) D(x, l) + E(x, l) & D(x, l)\ J(x)^T \\ J(x) D(x, l) & 0 \end{pmatrix} \begin{pmatrix} \bar{s} \\ \Delta l \end{pmatrix} = -\begin{pmatrix} D(x, l)\ \nabla_x l(x, l) \\ C(x) \end{pmatrix} \quad (3.96)$$

The form of the coefficient matrix in Expression 3.96, Assumptions 1, and Proposition 2, together imply that the affine-scaling interior-point Newton algorithm is well defined in a neighborhood of a non-degenerate regular point that satisfies the second-order sufficient optimality conditions.

For father description the next notation will be necessary

$$s = \begin{pmatrix} s_y \\ s_u \end{pmatrix},$$

(3.97)

$$D(x, l) = \begin{pmatrix} D_y(x, l) & 0 \\ 0 & D_u(x, l) \end{pmatrix}, \quad E(x, l) = \begin{pmatrix} E_y(x, l) & 0 \\ 0 & E_u(x, l) \end{pmatrix}, \quad (3.98)$$

and

$$\nabla_x l(x, \lambda) = \begin{pmatrix} \nabla_y l(x, \lambda) \\ \nabla_u l(x, \lambda) \end{pmatrix} = \begin{pmatrix} \nabla_y f(x) + C_y(x)^T \lambda \\ \nabla_u f(x) + C_u(x)^T \lambda \end{pmatrix} \tag{3.97}$$

Using this it is possible to rewrite the Newton equations as follows

$$\begin{pmatrix} D_y(x, \lambda)^2 \nabla^2_{yy} l(x, \lambda) + E_y(x, \lambda) & D_y(x, \lambda)^2 \nabla^2_{yu} l(x, \lambda) & D_y(x, \lambda)^2 C_y(x)^T \\ D_u(x, \lambda)^2 \nabla^2_{uy} l(x, \lambda) & D_u(x, \lambda)^2 \nabla^2_{yy} l(x, \lambda) + E_u(x, \lambda) & D_u(x, \lambda)^2 C_u(x)^T \\ C_y(x) & C_u(x) & 0 \end{pmatrix}$$

$$\bullet \begin{pmatrix} \begin{pmatrix} s_y \\ s_u \end{pmatrix} \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} D_y(x, \lambda)^2 \left(\nabla_y f(x) + C_y(x)^T \lambda \right) \\ D_u(x, \lambda)^2 \left(\nabla_u f(x) + C_u(x)^T \lambda \right) \\ C(x) \end{pmatrix} \tag{3.98}$$

From the first linear Equation 3.98 it is possible to obtain

$$\Delta\lambda = -\left(C_y(x)^{-T} \nabla^2_{yy} l(x, \lambda) + C_y(x)^{-T} D_y(x, \lambda)^{-2} E_y(x, \lambda) \quad C_y(x)^{-T} \nabla^2_{yu} l(x, \lambda) \right) \begin{pmatrix} s_y \\ s_u \end{pmatrix} - C_y(x)^{-T} \nabla_y f(x) - \lambda \tag{3.99}$$

The third linear equation can be rewritten as

$$C_y(x) s_y + C_u(x) s_u = -C(x) \tag{3.100}$$

since $C_y(x)$ is nonsingular, it is possible to obtain

$$s = s^n + W(x) s_u \tag{3.101}$$

where

$$s^n = \begin{pmatrix} -C_y(x)^{-1} C(x) \\ 0 \end{pmatrix} \tag{3.102}$$

is a particular solution of the linearized state equation, and

$$W(x) = \begin{pmatrix} -C_y(x)^{-1} C_u(z) \\ I_{n-m} \end{pmatrix}$$

(3.103)

is a matrix whose columns form is a basis for the null space of the Jacobian matrix J(x).

The second linear Equation 3.98 can now be rewritten only in variables s_u by using the Formulas 3.100, 3.101, 3.102, and 3.103 from above.

$$\left(W(x)^T H^a(x, \lambda) W(x)\right) s_u = -W(x)^T \left(H^a(x, \lambda) s^n + \nabla f(x)\right), \tag{3.104}$$

where

$$H^a(x, \lambda) = \begin{pmatrix} \nabla^2_{yy} l(x, \lambda) + D_y(x, \lambda)^{-2} E_y(x, \lambda) & \nabla^2_{yu} l(x, \lambda) \\ \nabla^2_{uy} l(x, \lambda) & \nabla^2_{uu} l(x, \lambda) + D_u(x, \lambda)^{-2} E_u(x, \lambda) \end{pmatrix} \tag{3.105}$$

is an augmented of the Hessian $\nabla^2_{xx} l(x, \lambda)$ of the Lagrangian function l(x,ë). The augmented term

$$G(x, \lambda) = \begin{pmatrix} G_y(x, \lambda) & 0 \\ 0 & G_u(x, \lambda) \end{pmatrix} = \begin{pmatrix} D_y(x, \lambda)^{-2} E_y(x, \lambda) & 0 \\ 0 & D_u(x, \lambda)^{-2} E_u(x, \lambda) \end{pmatrix} \tag{3.106}$$

takes into account the presence of the bound constraints in the variables y and u.

Affine-scaling interior-point Newton algorithm. In summary, the affine-scaling interior-point Newton algorithm is the following:

1. Choose an initial point (x,ë) with x > 0
2. Until convergence do

2.1a Compute $(s, \Delta\lambda)$ by solving Equation 3.96 for $(\bar{s}, \Delta\lambda)$ and then by setting $s = D(x, \lambda)\bar{s}$ or, equivalently,

2.1b Compute s^n as in Equation 3.102.

Compute s_u by solving Equation 3.104)

Compute $s_y = (s^n)_y - C_y(x)^{-1} C_u(x) s_u$

Compute $\Delta\lambda$ by Equation 3.99

2.2 Set $\alpha = \dfrac{-1}{\min(X^{-1}s, -1)}$, where $\tau \in (0,1)$.

Set the new iterate (x,ë) to (x,ë)+$(\alpha s, \Delta\lambda)$.

This algorithm is locally well-defined around a point satisfies the standard Newton assumptions, i.e., a non-degenerate regular point for which the second-order sufficient optimality conditions hold.
The q-quadratic rate of convergence of this algorithm is guarantee by the next Theorem.

Theorem 5. *Lets Assumptions 1 hold and consider a sequence in the pair (x,ë) that satisfied the second order sufficient optimality conditions. If ô is chosen so that* $|t-1| = O\left(\|F_1(x, l)\|\right)$, *where*

$$F_1(x, l) = \begin{pmatrix} D(x, l)^2 \nabla_x l(x, l) \\ C(x) \end{pmatrix},$$

then a sequence converges with a q-quadratic rate. The corresponding sequence in z generated by z=z(x,ë) converge r quadratic to $z_* = z(x_*, l_*)$.

Conclusions. The algorithm for solution of the control problem for airflow in duct networks as the mathematical programming task is outlined according to Vicente (1996). This algorithm can be used as a base for computer code development.

3.4 Final discussion

Different approaches for solving systems of nonlinear equations was analyzed. It was shown that the more promising approaches are the family of Newton Affine Invariant methods, the Tensor methods and the Homotopy methods.

If a good initial solution is known, the best choice is one of the Newton Affine Invariant methods. For comparatively large problem, when time for solution is significant, the Tensor methods are a good choice.

For the most practical cases, when either: (1) time is deficient to search for a good initial solution, (2) automation is needed for different types of evaluation of technical solutions, or (3) parametric evaluation is necessary, Homotopy methods are the best.

All methods shall included into the computer code. The first version of the code should be based on the Homotopy method as the most user friendly and robust. Moreover, Newton Affine Invariant methods and Tensor methods can be used as a part of Homotopy method. In this case Homotopy method will be a conceptual basis for solution.

4. Problem formalization

The proposed definition of the problem considers laboratory rooms, as well as supply and exhaust air distribution systems, fume hoods, exfiltration and infiltration, fume hood fans and filters, and other auxiliary fans as nodal parts of a network.

At any time the mass of air entering the room is always equal to the mass of air leaving this room. This includes air entering the room with infiltration or exiting the room with exfiltration. In spite of this equality, laboratory spaces are always under either positive or negative pressure. Pressure in the room depends not on the difference between supply and exhaust air but on the position of "zero-point", which is the location where the positive pressure originated by supply system changes to the negative pressure originated by the exhaust system. This position is a function of central supply and exhaust fans, resistance of ducts and fittings, position of dampers, duct and construction leakage, open/closed doorways, and performance of unitary, auxiliary, and/or manifolded fans. Doorways are assumed closed. However air is leaking through door slits. Schematically, if a "zero-point" is "after" the room, the room pressure is positive, and if "zero-point" is "before" the room, the room pressure is negative. The easiest way to control "zero-point" is by changing the resistances of dampers either in supply or in exhaust/return duct connected to the particular room.

It will be shown that the air conditioning systems under investigation are not the tree-networks studied by the T-Method (1990) but a truly free topology 2-dimensional network with loops. Even a part of this network, the central supply and the exhaust systems, can be associated with tree-network sub-systems.

The major difference between existing methods and the proposed one is that in the proposed method the multi-fan multi-lab network simulation coexists with control system simulation and statistical modeling.

4.1 Topology

Topologically laboratory air distribution systems which include central supply, central exhaust systems, many exhaust fans, rooms, and in/exfiltration can be represented by a cycle graph system. Topology of such systems can be simplified when it is considered that a subtree represents the central supply system where the supply fan(s) is the root. The central laboratory exhaust system is sometimes topologically isomorphic to the central supply system. Such isomorphic supply and exhaust systems can be represented by two tree-graphs connected at the terminals (Figure 3). Every root of such a tree-graph includes a supply or an exhaust fan or a number of fans connected in parallel (often) or in series (seldom), and every terminal is a connection to an internal laboratory space.

The most important benefit of the isomorphic topology is that the systems have a "limited cycle structure". The main cycles of such systems can be simplified by "gluing" two tree-graphs at their hanging nodes (leaves). This allows one to identify the flow directions in the common arcs from supply to exhaust avoiding the main difficulties of calculating the systems with cycles. The only difficulty with identifying flow directions is the infiltration/exfiltration element since the pressure is unknown at the beginning of calculation. However, this element of the network is always belongs to the branches.

The following four schematics of air distribution systems demonstrate the transformation of an air network into a graph. All four schematics are only examples of topology formalization and do not pretend to be a part of engineering design.

Figure 4 presents a constant volume supply and exhaust system with parallel fans and reheat coils controlling temperatures in spaces #1, #2, and #3. The formalized topology of this system is presented on

Figure 5. Two nodes present each network element; one of which is an inlet and another is an outlet.

Pressure difference between these nodes represents the pressure loss in the element. The exception to this rule is the space/room, which assumes no pressure difference. Reheat coils control the temperatures in the rooms. The use of constant volume terminals assumes that the pressure in the spaces is kept the same as surrounding dampers BD1 through BD6 set it.

Figure 6 presents a variable air volume system with infiltration/exfiltration element in the space #3 and exhaust stack. In this example maximum concentration of gasses at air discharged from the exhaust stack is limited. This is warranted by using an air bypass damper CD4 sensoring air velocity on the top of the stack as shown in Figure 6. The supply fan curve is controlled by the static pressure sensor located in the supply duct. This sensor sets variable frequency drive VFD1 into a position that keeps static pressure in the supply duct constant. Variable air volume terminals together with exhaust dampers CD1, CD2, and CD3 control pressure and temperature in the rooms. This schematic is formalized on Figure 7.

Figure 8 presents a variable air volume system with air cascading and two fume hoods located in the rooms. The variable volume terminal VAVT1 controls air flow supplied to the corridor. Air flow from the corridor #1 is entering the rooms through cascading dampers C1 and C2. Auxiliary fan F4 supplies air to the fume hood AFH1. There are two exhaust ducts in this room, one from the fume hood and the second from the room itself. The second exhaust is balancing the room pressure when the fume hood is changing the regime. The second fume hood in room #3 has an individual filter FL3 and individual exhaust fan F3. The pressure in the room is controlled by the damper D3 similar to the room #2. This schematic is formalized on Figure 9.

Figure 10 presents a variable air volume system with manifold fume hood system and supply and exhaust plenums. This system is equipped with computer that analyzes the pressure and temperature in each room and controls the variable frequency drive VFD2 at central exhaust fan. There are three fume hoods: one in the room #1 served by the unitary exhaust fan F3, two in the room #2 served by the two individual fans F4 and F5, and three in the room #3 served by the manifold system connected to the exhaust fan F6. There is an additional pressure balancing exhaust fan F2 in the room #1. Also this room has an infiltration/exfiltration element. This schematic is formalized on Figure 11. Even for this complicated system the direction of flow can be easily identified for all arcs except the infiltration/exfiltration element in room #1.

Figure 12 is similar to the previous Figure 10 however it does not have the central exhaust system since the design amount of exhaust air is higher than necessary for ventilation. This is very often practical design case.

Figure 4. Constant Volume Central Supply and Exhaust Reheat Systems

Figure 5. Formalized Topology for Constant Volume Central Supply and Exhaust Reheat Systems

Figure 6. Variable Air Volume Central Supply and Exhaust Systems

Figure 7. Formalized Topology for Variable Air Volume Central Supply and Exhaust Systems

Figure 8. Variable Air Volume Central Supply and Exhaust Systems with Cascading and Fume Hoods

Figure 9. Formalized Topology for Variable Air Volume Central Supply and Exhaust Systems with Cascading and Fume Hoods

Figure 10. Variable Air Volume Central Supply and Exhaust Systems with Plenums, Manifold Exhaust, and Computer Control.

Figure 11. Formalized Topology for Variable Air Volume Central Supply and Exhaust Systems with Plenums, Manifold Exhaust, and Computer Control.

Figure 12. Central Supply with Local and Manifold Exhaust.

Figure 13. Formalized Topology for Central Supply with Local and Manifold Exhaust.

4.2 Control function

Control of fan discharge or resistance of dampers, terminal boxes, and sashes depends on location on their sensors. The user will be able to locate these sensors in any place of the system. Such control parameters will be:

-- Static pressure,
-- Total pressure,
-- Air flow volume,
-- Air flow volume difference,
-- Air velocity,
-- Static pressure difference,
-- Air temperature.

As one can see, in case of air flow or static pressure differences, two sensors must be located. The user must present following data for each control device:

-- Control parameter,
-- Sensor locations (two locations for flow or pressure differences),
-- Initial setpoint,
-- Minimum-maximum range limitation,
-- Controller function.

Coefficients for a controller described in an algebraic polynomial equation will be a part of the user input. The graphical input of this function should be evaluated. It will transmit the signal received from the sensor to the control device as a response to a change in parameters. The possibility of simulating the automatic reset of control parameters and setpoints will be investigated.

Fan operating point will be controlled by one of the following devices selected by the user: (1) discharge or inlet damper, (2) inlet vanes, (3) pitch, or (4) variable frequency drive (VFD).

Similarly, it will be possible to control each damper from a sensor located in any place of the system selected by the user, including duct, room, cabinet, fan or exhaust stack discharge. For example, an outside air bypass damper in the exhaust plenum will start to open if the exhaust stack discharge velocity becomes smaller than the air velocity setpoint assigned by the user (mostly around 3000 fpm). Another simulation of control element (or exhaust fan) will be the bypass grille located in fume hoods.

4.3 Elements

In addition to a network, there is an important definition of system elements and system arcs. Arcs represent scalars of air flow. Most elements are represented as arcs between two nodes, low pressure and high pressure. Three arcs between four nodes (wye or tee) or four arcs between five nodes (cross) represent junctions. Some elements like entries, infiltration/exfiltrations elements, and exits are associated

with the terminal nodes. Others such as fans, dampers, fume hoods, BSC, many fittings are connected to the intermediate nodes. A special definition is made for rooms: a single node rather than an arc present a room. This means that a single pressure and a single flow characterize each room. The air resistance in the room is ignored and the flow is equal to the total supply or total exhaust air.

Arcs are mostly ducts and plenums. Infiltration into a room from the atmosphere or exfiltration from a room into the atmosphere is represented by an arc where the room and the atmosphere are pressure nodes. A junction that connects three ducts: main, common, and branch has three arcs with different flows and four nodes. The static pressure in the intermediate node will represent the average static pressure in the junction. Each system arc and node is associated with a number of parameters. Following is the list of major elements, arcs, and parameters:

Arc/node	Element name	Parameters
1-Arc	Duct section	Topological connections, Air flow, Shape, Size, Length, Temperature, Velocity, Pressure loss, Static pressure at both nodes, Leakage, Density, Viscosity, Reynolds number, Roughness, Friction coefficient, Sum of C-coefficients related to the section velocity, Heat loss/gain
2-Arc	Doorway slit at closed door	Topological connections, Air flow, Equivalent size, Velocity, Pressure difference, Static pressure at both nodes, Density, Viscosity, C-coefficient
3-Arc	Infiltration/ Exfiltration	Topological connections, Air flow, Equivalent size, Velocity, Pressure difference, Density, Static pressure at both nodes, Viscosity, C-coefficient
4-Arc	Plenum (pressure loss is neglected)	Topological connections, Air flow, Shape, Size, Length, Temperature, Leakage, Static pressure
5-Arc	Heating/Cooling Coil	Topological connections, Air flow, Heating/cooling capacity, Shape, Size, Temperature at both nodes, Average velocity, Pressure loss, Static pressure at both nodes, Density at outlet, Viscosity, C-coefficient
6-Node	Zone/Room	Topological connections, Static pressure, Temperature, Air flow, Heat loss/gain
7-Node	Dampers: fire/smoke/back draft	Topological connections, C-coefficient at open position, Open/close position
8-Node	Dampers: control or balancing	Topological connections, Type of damper and associated geometry, Angle, C-coefficient

9-Node	Fittings: Entry Exit Transition Junction Obstruction Louvers Fan/duct link	Topological connections, Geometry, Size, C-coefficient, Other information associated with the type of fitting (ASHRAE Handbook 1995, Chapter 32 "Duct Design" or DFDB (1994)
10-Node	VAV control box	Topological connections, Pressure/temperature setpoint, C-coefficient
11-Node	Constant volume box	Topological connections, Volume flow setpoint, C-coefficient
12-Node	Fan	Topological connections, Type of fan, Fan curve, Inlet/outlet size, Density at inlet, Heat gain, Type of control, RPM, Efficiency, Brake power, Inlet vanes percent of opening, Pitch angle
13-Node	Hood/canopy island	Topological connections, Type, Size, C-coefficient
14-Node	Sound attenuator	Topological connections, C-coefficient
15-Node	Fume hood or biological safety cabinet	Topological connections, Maximum sash and bypass free areas, Function between free areas of sash and bypass, C-coefficient, Operating schedule

The relationship between hydraulic resistance of fittings or equipment and actuators position is described by C-coefficients (DFDB 1994). These coefficients are mostly presented in a table form, however for some of them algebraic formulas exist (Idelchik 1996). In our opinion, the more general way of using C-coefficients is the tables.

Following is the list of hydraulic equations for the main elements recommended for Bellair computer program (units see above):

(1) **Duct/Plenum**.

Darcy-Weisbach equation

$$\Delta P = \left(\frac{fL}{D_f} + \sum C \right) \frac{V^2 \rho}{2 g_c} \tag{4.1}$$

Equivalent-by-friction diameter for rectangular ducts:

$$D_f = 2 \frac{H \times W}{H + W} \tag{4.2}$$

Air density

$$\rho = 1000\frac{(P_s - 0.378P_W)}{287.1 + (273.15 + T)} \tag{4.3}$$

Reynolds number

$$\mathrm{Re} = \frac{D_h V}{\nu} \tag{4.4}$$

For fully developed laminar-viscous flow when Re [2300

$$f = \frac{64}{\mathrm{Re}} \tag{4.5}$$

Altshul-Tsal's equation (ASHRAE 1997) for fully developed turbulent flow (Re >2300)

$$f' = 0.11\left(\frac{12\varepsilon}{D_h} + \frac{68}{\mathrm{Re}}\right)^{0.25} \tag{4.6}$$

if f' ≥ 0.018 then f = f'
if f' < 0.018 then f = 0.85 f' + 0.0028

Air leakage

$$\Delta G = a_1 C_L P_s^{0.65} \tag{4.7}$$

Heat loss/gain from/to a duct (ASHRAE 1997)

$$Q = \frac{UPL}{(cf)(T_{av.duct} - T_{ambient})} \tag{4.8}$$

where:

cf = converging factor, 1000 for SI units [12 for IP units]
U = duct overall heat transfer coefficient, W [Btu/h-ft^2-F]
P = perimeter of duct, m [in]
L = duct length, m [ft]

(2) **Doorway slips**. The technique for calculating infiltration and exfiltration presented in ASHRAE 1997 Handbook is based on Darcy-Weisbach equation solved in respect to flow:

$$G = C_D A\sqrt{\frac{2\Delta P}{\rho}} \tag{4.9}$$

where:

G = Air flow rate, cfm [m^3/s]

C_D = Discharge coefficient for opening, dimensionless
A = Cross-section area of opening, ft^2 [m^2]
ÄP = Pressure difference across opening, in.WG [Pa]
ñ = Air density, lbm/ft^3 [kg/m^3]

The relation between the discharge coefficient C_D and local resistance coefficient C is

$$C_D = \frac{1}{\sqrt{C}} \tag{4.10}$$

The local resistance coefficient for a thick-edged orifice is presented by Idelchik (1996, p.219, diagram 4-12):

$$C = 0.5\left(1-\frac{A_0}{A_1}\right)^{0.75} + \left(1-\frac{A_0}{A_2}\right)^2 + \tau\left(1-\frac{A_0}{A_1}\right)^{0.375} \times \left(1-\frac{A_0}{A_2}\right) \tag{4.11}$$

where:

A_0 = Free orifice area, ft^2 [m^2],
A_1 = Wall area from inlet side, ft^2 [m^2],
A_2 = Wall area from outlet side, ft^2 [m^2],
ô = Function which depends on the ration between the thickness of orifice to its hydraulic diameter, [from 0 to 1.3, average=1]

$$\tau = (2.4-\bar{l})\times 10^{-j}, \qquad j = 0.25+\frac{0.535\bar{l}^8}{0.05+\bar{l}^8}, \qquad \bar{l} = \frac{l}{D_h}$$

l = Orifice thickness, ft [m]

Assuming very large wall areas and low door thickness having $\bar{l} = 0.2$ and τ = 1.22, coefficients C is 2.72 and C_D=0.606

(3) **Infiltration/exfiltration**. Similar to doorway slips.

(4) **Plenum**. Air leakage similar to duct leakage related to plenum surface and depends of leakage class.

(5) **Heating/cooling coil**. Unless C-coefficient is given, coil resistance is presented as pressure loss for nominal flow. For simulation reason pressure loss has to be rearranged into C-coefficient:

$$C = 2\frac{\Delta P\, A^2}{G^2\, \rho} \tag{4.12}$$

where:

C = Local resistance coefficient, dimensionless
ÄP = Pressure loss at nominal flow, Pa [in.WG]
G = Flow, m^3/s [cfm]
A = Coil cross-section area, m^2 [ft^2]
ñ = Air density, kg/m^3 [lbm/ft^3]

The outlet coil temperature is calculated as (simplified case):

$$T_{out} = T_{in} \pm \frac{Q}{(cf)\, c_p\, \rho\, G} \tag{4.13}$$

where:

cf	= Unit converging, 1.0 for SI (60 for IP)
T_{in}	= Coil inlet temperature, °C [°F]
Q	= Heating/cooling load, kJ [Btuh]
c_p	= Specific heat at constant pressure, kJ/(kg-°K)[Btu/lb_m-°F]

The function of the controller changing heating/cooling load can be approximated as a polynomial

$$Q = aT_r^{\,3} + bT_r^{\,2} + cT_r + d \tag{4.14}$$

where:

T_r	= Room temperature, °C [°F]
a,b,c,d	= Constants

(6) **Room**. It was mentioned above that pressure loss in a room could be neglected. However, static pressure in a room and its sign that depends on position of "zero pressure" point is one of the most important design parameter.

Thermal regime in a room is a function of heat balance that is the difference between heat loss and heat gain. The temperature in the room is

$$T_r = T_{in} \pm \frac{\sum Q}{(cf)\, c_p\, \rho\, G} \tag{4.15}$$

where:

cf	= Unit converging, 1.0 for SI (60 for IP)
T_r	= Room temperature, °C [°F]
ÓQ	= The sum of heat gain/loss, kJ [Btuh]
c_p	= Specific heat at constant pressure, kJ/(kg-°K)[Btu/lb_m-°F]

The variable ÓQ includes of heat gain/loss from walls, ceiling, floor, doors and windows, partitions, infiltration/exfiltration, people, equipment, appliances, electrical motors, and lights.

(7) **Fire/smoke/backdraft damper.** The local resistance C-coefficient for open damper is presented in SMACNA book (SMACNA 1995).

(8) **Control damper.** The local resistance C-coefficient is presented in DFDB (1994) as a function of angle, shape, type (butterfly, gate, parallel or opposed blades), number of blades, ratio of open area (for gate), duct perimeter and blades length (for multi-blades).

(9) **Fittings.** C-coefficients for fittings can be obtained from DFDB (1994). Many tables, one-, two-, and

three-dimensional represents them. These tables have to be interpolated either ones at the beginning of calculation (fixed fittings) or at each iteration (variable fittings).

(10) **VAV control box**. C-coefficient should be obtained from manufacturers.

(11) **Constant volume box**. C-coefficient should be obtained from manufacturers.

(12) **Fan**. As mentioned above there is a number of ways to represent the flow-pressure characteristics of a fan. The easiest way is to approximate the curve by a few lines. However, the best calculation results can be obtained when the curve is approximated by a formula. This should avoid oscillations at calculation. The formula recommended by Stoecker (Stoecker 1975) is proposed:

$$Q_f = c_1 + c_2 P_f + c_3 P_f^{\,2} + c_4 N + c_5 N^2 + c_6 P_f N + c_7 P_f^{\,2} N + c_8 P_f N^2 + c_9 P_f^{\,2} N^2 \qquad (4.16)$$

The coefficients c_1 through c_9 should be approximate automatically in a computer program for a representative curve and recalculated for any other rotation using the "Fan Laws" (ASHRAE 1996)

$$Q_{f_1} = Q_{f2} \frac{N_1}{N_2} \qquad (4.17)$$

$$P_{f_1} = P_{f2} \left(\frac{N_1}{N_2} \right)^2 \qquad (4.18)$$

(13) **Hood/Canopy**. C-coefficients for fittings can be obtained from DFDB (1994).

(14) **Sound attenuator.** C-coefficients can be obtained from DFDB (1994) as a part of duct mounted equipment.

(15) **Fume hood/Biological safety cabinet.** C-coefficient, which for many types of fume hoods depends on sash position, should be obtained from manufacturers.

5. Main flow chart

The air distribution systems in a large laboratory building will be divided into a number of units simulated separately. It is proposed that each calculation unit will consist of only one central supply system with many corresponding exhaust systems, individuals, manifolded, centralized or combined.

The proposed numerical method will perform four major calculations:

(1) Definition of air flows and pressures for known setpoints of control devices (damper openings, given fan RPM, known opened area for sashes, etc.),

(2) Definition of setpoints for control devices,

(3) Random selection of the positions of FH/BSC sashes,

(4) System performance analysis.

As a result of such calculation the user will receive air flows, air velocities, in all system sections and pressures in all system nodes. This includes workspaces, manifolds, fume hood inlet sashes, stack exhausts, and ductworks. Also, the user will receive fan operating points and fan characteristics, positions of all system control devices, and corresponding electrical energy consumption. Air system performance will be studied in the third step by randomly selected positions of sashes and other control and laboratory units. The user will input the loading schedule for each unit and computer will randomize the position of control device and create a total usage (diversity) factor as a recommendation for optimum design. There are four major fragments of the program Bellair:

(a) Preprocessor that includes data input, verification, and printout,
(b) Solver that includes calculation routines,
(c) Postprocessor that includes verification and printout of the results,
(d) Database population and access.

Following is the flow chart of the modeling process:

(1) Preprocessor. In the flow chart (Figure 14) preprocessor is presented by a single block. This block consists input data entry, including topology, duct size, fitting, fans, terminals, infiltration/exfiltration elements, fume and BSC schedules, sensors, controllers and actuators, allowable parameter ranges in the rooms, allowable velocity range in the opening of each sash, and other equipment. One of the main parts of the preprocessor is data evaluation.

(2) Local Fitting Libraries. Computerized fittings selection and C-coefficients interpolating from tables at each iteration takes substantial computer time for selecting the fittings from the global library on a hard disk. A speed up process of creating and using local libraries was proposed in T-Duct (Tsal et.al. 1990) and will be used in Bellair. It is base on the principle that fittings at the same systems are mostly identical. All fittings are divided into two types: fixed and variable. Fixed fittings are not changing C-coefficients with iteration and can be calculated once only. Variable fittings must be calculated at each iteration. At the beginning the program copies all fittings necessary for the calculated system from the global fitting library on a hard disk. Then it analyzes each fitting and, if the fitting is fixed, it calculates the C-coefficients and stores only its value into a local library of fixed fittings for future use. If the fitting is variable, it stores the selection table (or formula) into a different small local library of variable fittings on RAM. No more selection is needed from the global library on hard disk. According to T-Duct, this procedure drastically reduces the CPU time and makes unnecessary the use of approximation formulas for fittings. Notice that the resistance of FH/BSC sashes depends on its free area opening and belongs to the variable fittings.

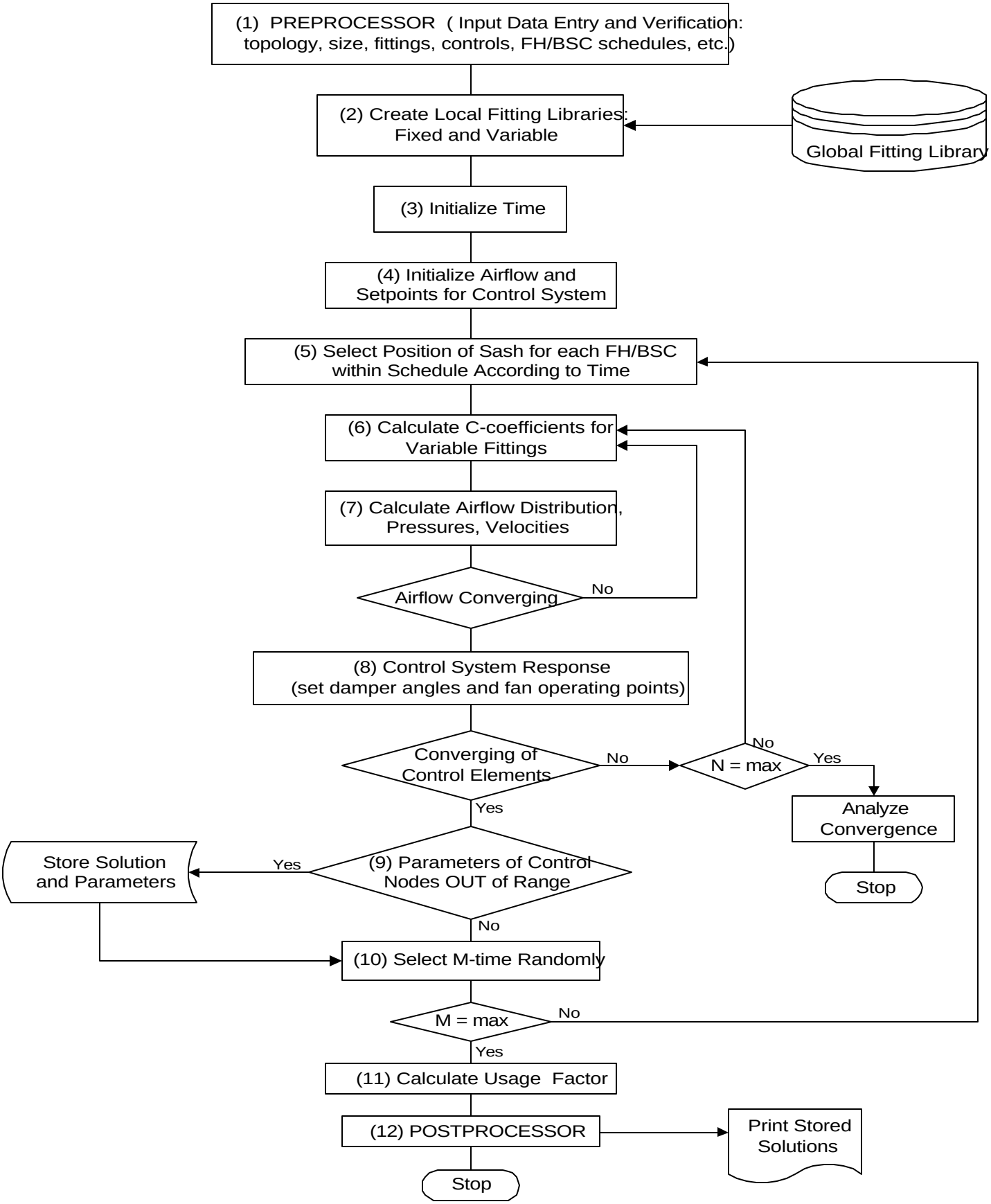

Figure 14. Flow Chart of the Computer Program Bellair

(3) Initialize Time. Proposed computer program will include statistical analysis of sash positions that depend on the time of the day and on the FH/ BSC schedules. The third block of the flow chart initializes the time of the day used later to access the fume hood schedules.

(4) Initialize Air flow and Setpoints. In this block the air flow at each terminal arc will be initialized based on the average air velocity (for example, 7 m/s). The air flows for the common duct sections will be calculated. Then the setpoints of each damper and VFD will be initialized.

(5) Position of Sashes. Using the time of the day selected in the block 2 and the schedule for a particular FH/BSC the program will select the position of the sash.

(6) Variable Fittings. Computation of C-coefficients has to be performed for variable fittings only using the local fitting library located in RAM. These calculations use flows, velocities, and Re coefficients which are changing at each iteration. Also, calculation is performed at this block for FH/BSC C-coefficients based on the sash position selected in the previous block.

(7) Air flow Distribution. This block represents the major part of the solver that is the calculation of air flow distribution based on pressure and air mass balancing equations using the one of the methods explained above (Newton, Tensor, or Homotopy). The results of this calculation is the air flows, velocities, and temperatures at each duct section, and pressures at each node, as well as fan flows and total flows passing each element, including rooms. The true air flow distribution can be achieved as a result of iteration process. Therefore, after block 7 has been executed, there is a check for convergence and a return to the block 6 if convergence has not yet been achieved.

(8) Control System Setpoints. In the previous block air flow calculation was based on the initialized setpoints. In this block setpoints will be assigned for each controller based on the actual air flows, pressures, pressure differences, and velocities calculated in the block 7, and, after checking for the convergence, calculating process will return to block 6. Convergence at this step means that, as results of iteration, there is no significant change in setpoints for all control devices.

There is a possibility that, in spite of many iterations, converging does not occur after a large number of iterations (N = max). This can happen because of the overdefined control system that produces unlimited number of solution. The computer will detect such case and execution will be terminated in order to give user an opportunity to analyze the system and to cure the problem.

(9) Range Checking. At previous steps air distribution is obtained for initially selected time of the day at randomly opened sashes. At this time air flow, pressure difference, and temperature are checked at each room and air velocity is checked at each sash and at stack discharge. If these parameters are out of the minimum-maximum range presented in input data this case is considered as a violation and is stored on the hard disk for further investigation. If there is no violation the execution will proceed to the next block.

(10) Time Selection. Next time is randomly selected. If the number of selections does not exceed the maximum the process is returning to the block 5 for selecting the position of the sashes. Otherwise calculation moves to the next block.

(11) Usage (diversity) Factor. Diversity factor is calculated for the supply system under investigation.

This block includes the Postprocessor.

(12) Postprocessor. Postprocessor displays and prints all results of calculation.

6. Principles of software development

Computer program Bellair will be divided into five major parts:

(1) GUI preprocessor
(2) Global data base
(3) Local data base
(4) Solver
(5) Postprocessor

The main principle of the program architecture is the modular approach using object oriented programming language C++. The most possible user-friendly input will be developed. On-line data checking is one of the most important parts of the preprocessor.

A flexible units selection system should be implemented where the user will be able to select individual units from any unit system, as well as recommended SI, IP, and Metric systems.

Graphically entered topology is planed for the second version of the code. In the first version numerical linkage between the nodes is proposed. However, selection of fittings and elements should be base on their graphical representation.

The most difficult part of input data is fitting selection. In order to simplify fitting selection from a local fitting library this part of the preprocessor will use drag-and-drop techniques with fitting pictures appearing on the screen.

The second kind of simplification will be the preselection of fittings prior to their appearance on the screen similar to the technique presented in T-Duct. Only applicable fittings will appear for each section. This selection will be based on the following four conditions:

-- Exhaust/return or supply system. Only fittings that belong to the required system are selected. For example, exits will not be selected at all for exhaust and entries will no be selected for supply,

-- Duct shape. Only fittings with the same shape as calculated duct section would be displayed,

-- Parent shape. Only fittings having the parent shape the same as parent for a calculated duct section will be selected. For example, if a duct section is round but its parent is rectangular, the junction fittings will be selected only having rectangular shape with round parent.

-- Parent size. C-coefficient for junction depends on size ratio between parent and children

sections. Program will preselect only those junctions that satisfy the section-parent size conditions: (1) section size equal its parent size or (2) section part is less size than its parent size.

One of the most common mistakes made by a user is the selection of different junctions for the same node during straight and branch sections data input. This will be prevented by automatic assignment of the same junction to the linked section.

Similar to T-Duct, the most important variable will be mass flow not volumetric flow. Therefore, the volume of flow entering a fan will never be equal to the flow volume at fan discharge.

Statistical analysis will be conducted either by a number of searches assigned by the user, maximum computer time, or the standard deviation difference.

7. Software implementation plan

The main platform for the BELLAIR code is MS Windows 95/98/NT for IBM PC computers or compatibles. Programming language is C++, recommended compiler is Visual C++, version 5.0 or higher. DEMO of the program Bellair will be installed on Internet.

Following steps has to be conducted for development and implementation of computer code:

Preprocessor consists main interaction routines between the user and software. The program must be developed user-friendly considering unexperienced user having knowledge in duct design but not in duct simulation. Fitting selection will be organized using drag-and-drop technique. Data checking should be if possible checked while data entry and the messages should describe the error as well as present recommendation for its correction. The messages should be divided into two classes: fatal and warning. It is necessary to have an opportunity to return to any previous screens. Full capability of insert/copying/moving/deleting any part of the data should be implemented using data blocking. The most common user mistake, which is selection of different junctions for the same node, has to be carefully diagnosed and accurately explained. Help screens should includes examples. Graphic representation and selection shall be used for fitting selection. Graphic input for topology and length has to be investigated. Following are the steps for developing the preprocessor:

- Architecture
- Topology representation (graphical and numerical)
- Graphic selection of fittings
- Local fitting data base real time development
- Input data variables and their units
- Screens and menus
- Elements and fittings graphics
- Data checking and diagnostic
- On-line error messages
- Data manipulation routines (browse/insert/copy/move/delete)

- Information accessed from the database
- Help index, screens, and messages
- Input data listing
- Classes

Data Base must be developed for fittings and equipment, roughness, specification and cost data.

- Fittings development and population
- Duct roughness
- Elements data
- Program messages (data driven information)
- Graphic representation of fittings and elements

Solver consists all calculation routines. It will include major three parts: hydraulics, control, statistics. The hydraulics part will consist three following methods: Newton, tensor, and homotopy. The selection of a method will be performed automatically by the program. Following are the development steps:

- Architecture
- Algorithm
- Local fitting library access
- Element data access
- Fitting tables access and approximation
- Duct hydraulics
- Heat loss and air leakage
- Newton method calculation
- Tensor method calculation
- Homotopy method calculation
- Control equipment simulation
- Statistical data selection and recording
- Classes
- Output data storing

Postprocessor will be developed for displaying and printing the results. It will be linked to the Preprocessor which will allow to perform interactive studies. The development of Postprocessor includes:

- Output screens
- Graphical fan-system characteristics
- Graphical pressure diagrams
- Hard listing of the results
- Classes

Installation program will allow user to install software on his computer.

- Program installation routines using WISE program

Demo development will include

- Algorithm
- Screens
- Subroutines
- Main program
- HTML programming
- Internet downloading

Manual has to include following sections:

- Abstract
- License and disclaimer
- Program overview
- Hardware requirements
- Program installation
- Quick start
- Practicing with program
- Run and output
- Examples
- Data verification and validation

Testing

- Alpha testing
- Beta testing
- Code correction and modification

Publicity

- Articles publication
- Presentations at conferences
- Teaching seminars
- User group

Program support is one of the most important part of the implementation plan. In spite of the fact that development of the software is plan to be budgeted, a support should be profit based. It includes:

- Perform software house functions
- Telephone support
- E-mail/fax written support

- Update information

8. Conclusion

The purpose of the project is to develop and implement a practical tool, a computer code that can be used by HVAC engineers in studying, designing, and retrofitting of air distribution systems in research laboratories.

The proposed computer code, named Bellair, will allow HVAC engineers to calculate actual airflow, pressures, and fan operating points in laboratory multi-fan air distribution systems as well as static pressure in laboratory spaces at different operating conditions where space pressurization, confinement zoning, and flow/pressure stability are the most important requirements. Also, the computer code will be capable of modeling the control of each fan or damper from a sensor located in any place of the system specified by the user including ducts, laboratories, cabinets, exhaust stack discharge, etc. The control parameters could be static pressure, total pressure, airflow volume, air velocity, temperature, and pressure or temperature difference.

There are four major steps of multi-fan system simulation:

(1) Defining air flows and pressures for initially set control devices,
(2) Adjustment of positions of control devices in accordance with their set-up,
(3) Random selection of the positions of fume hood/BSC's sashes based on their loading schedules,
(4) System performance analysis.

Network simulation problem requires a solution for a large system of simultaneous nonlinear algebraic equations. Numerous methods capable of solving nonlinear algebraic equations for both branched and cycled networks simulation are studied. The most promising approaches for cycle systems are the family of Newton Affine Invariant methods, the Tensor methods, and the Homotopy methods. If a good initial guess is known, the best choice is one of the Newton Affine Invariant methods. For a comparatively large problem, when time for solution is significant, the Tensor methods are a good choice. For the most practical cases, when either: (1) time is deficient to search for a good initial guess, (2) automation is needed for different types of technical solutions evaluation, or (3) parametric evaluation is necessary, Homotopy methods are the best. All three approaches are expected to be included into the computer code and selected automatically during the calculation.

All methods for solving nonlinear equations involve the solution of many sets of linear algebraic equations. It is possible for small tasks to use simple methods based on direct solution techniques. Whenever there is sparse matrix, sparse mode elimination techniques may be successfully used. For large and super large tasks the Krylov's method expected to be applied.

Control system simulation requires the use of mathematical programming technique to get the minimization of an objective function that describes residuals. The Vicente's algorithm is suggested for solving this problem.

As a result of such calculation user receives airflow, pressures, and velocities at all system sections and pressures at all system nodes. This includes workspaces, manifolds, fume hood inlet sashes, stack exhausts, and ductworks. User also receives fan operating points, position of all system control devices, and electrical energy consumption. Air system performance planed to be studied by randomly selected positions of sashes and other laboratory units. As soon as user inputs the loading schedule for each unit, computer randomizes the position of sashes/units and calculates the total usage (diversity) factor as a recommendation for optimum design.

There are four major fragments of the Bellair program:

(1) Preprocessor that contains of data input, verification, and printout,
(2) Solver that includes calculation routines,
(3) Postprocessor that performs verification and printout of the results,
(4) Populated database.

Following are the main steps of development and implementation of the Bellair program:

(1) Create the program architecture,
(2) Perform coding and debugging,
(3) Provide efficient testing,
(4) Develop and populate the data base
(5) Develop Demo and install on the Internet,
(6) Write the user manual,
(7) Provide program installment, update, and support.

The recommended platform for the Bellair code is MS Windows 95/98/NT for IBM PC computers, programming language is C++, recommended compiler is Visual C++.

9. References and Bibliography

AABC 1965. Principles of Air Distribution, Los Angeles, Associated Air Balancing Council.

AABC 1983. Duct Leakage and Air Balancing. Technical Publication No. 2-83. Washington, DC: Associated Air Balance Council.

Ahmed O., J.W.Mitchell, S.A.Klein 1998. "Experimental Validation if Thermal and Pressure Models in a Laboratory Simulator", ASHRAE Transactions 1998, V.104, Pt.2

Allard, F. and Y. Utsumi 1992. "Air Flow through Large Openings", Energy and Buildings, Vol 18, p. 133-146

Altshul, A.D. and P.G. Kiselev 1975. Hydraulics and aerodynamics, 2d ed. Moscow: Stroisdat Publishing House.

Andriyashev, M.M. 1932. Water systems calculation technique, Moscow

AMCA 1973. Fans and Systems, Standard 201-73, Arlington Height, IL

Arioli, M., I.S. Duff, N.I. Gould, and J.K. Reid (1990). Use of the P^4 and P^5 algorithms for in-core factorization of sparse matrices, SIAM J. Sci. Stat. Comput 11, 913-927.

Arklin, H. and A. Shitzer 1979. Computer Aided Optimal Life-cycle Design of Rectangular Air Supply Duct Systems. ASHRAE Transactions, Vol.85, Part 1, pp. 197-213.

ASHRAE 1997. Fundamentals handbook. Chapter 32, Duct design. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., Atlanta, GA

ASHRAE 1996. HVAC Systems and Equipment Handbook, Atlanta, GA

ASHRAE 1995. ASHRAE Standard 120P, Laboratory methods of determining flow resistance of air duct and fittings, Atlanta, GA

ASHRAE 1995. HVAC Applications Handbook. Chapter 43. Sound and Vibration Control. Atlanta, GA

ASHRAE 1994. HVAC Duct Fitting Data Base, Research Project 574-RP, Atlanta, GA

ASHRAE 1993. ASHRAE Professional Development Seminars - Air System Design and Retrofit, Atlanta, GA

ASHRAE/SMACNA/TIMA 1985. Investigation of Duct Leakage. ASHRAE Research Project 308.

Axley, J. 1987. "Indoor air quality modeling phase II report." U.S. National Bureau of Standards Report NBSIR 87-3661.

Axley, J. 1988. "Multi-zone dispersal analysis by element assembly." Submitted to Building and Environment.

Altshul, A.D., and P.G. Kiselev 1975. Hydraulics and aerodynamics, 2d ed. Moscow: Stroisdat Publishing House.

ASHRAE. 1985. ASHRAE Handbook - 1985 Fundamentals, Chapter 33, "Duct design." Atlanta: American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc.

ASHRAE. 1989. ASHRAE Handbook - 1989 Fundamentals, Chapter 32, "Duct design." Atlanta: American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc.

Baccarat, S.A. 1987. “Inter-zone convective heat transfer in buildings: a review." ACME Journal of Solar Engineering, Vol. 109, May.

Baker, P.H. , Sharples, S., and Ward, I.C. 1987. "Air flow through cracks." Building and Environment, Vol. 22 No. 4. pp. 293-304.

Bellman, R.E. 1957. Dynamic programming. New York: Princeton University Press.
Belsky, J.S. 1991. Design HVAC Systems for Laboratories, Pharmaceutical Engineering, July/August, Vol.11, No.4

Bertschi, R.L. 1969. A Computer Program to Optimize the Life-cycle Cost of a Fan-duct System. Urbana: University of Illinois (thesis).

Besant, Robert W., and Johnson, Alan B. 1995. "Reducing Energy Costs Using Run-Around Systems." ASHRAE Journal, February

Bjorck A., and I.S. Duff. 1980. A direct method for the solution of sparse linear least squares problems. Linear Algebra and Its Applications, 34:43-67

Blevins, R.D. 1984. Applied Fluid Dynamics Handbook. New York; Van Norstrand Reinhold.

Bouricha A. 1992. Solving large sparse systems of nonlinear equations and nonlinear least squares problems using tensor methods on sequential and parallel computers. Ph.D. thesis, Computer Science Department, University of Colorado at Boulder

Bouwman, H.B. 1982. Optimal Duct System Design. TNO Research Institute for Environmental Hygiene, Indoor Climate Division, Delft, Netherlands.

Brooks, P.J. 1993. New ASHRAE Local Loss Coefficients for HVAC Fittings, ASHRAE Transactions, V.99, Pt.2

Brooks, P.J. 1995. Duct Design Fundamentals, ASHRAE Journal, April

Butakov, S.E. 1949. Aerodynamic of industrial ventilation. Moscow: Profstroisdat Publishing House.

Chow S.N., Mallet-Paret, and Yorke 1978. Finding zeros of maps: homotopy methods that are constructive with probability one, Math.Comp., 32 (1978), pp. 887-899.

Coad, W.J. 1985. Simplified sizing of pipes and ducts, HPAC, July

Coad, W.J. 1980. Energy Concepts of Air Handling Systems, HPAC, August

Colebrook, C.F. 1938-1939. "Turbulent flow in pipes with particular reference to the transition region between the smooth and rough pipe laws." Journal of the Institution of Civil Engineers, London, Vol 11.

Coogan J.J. 1996. " Effect of Surrounding Spaces on Rooms Pressurized by Different Flow Control" ASHRAE Transactions 1996, Vol.102, Pt.1

Cross, H. 1936. "Analysis of Flow in Networks of Conduits or Conductors", University of Illinois, Engineering Exp. Station, Bulletin #286, Urbana-Champaign

Data Research 1985. Utility Costs Forecasting, Data Resources Inc., Service Review, McGraw Hill Publishing House.

Davis T.A. and I.S. Duff (1995) A combined unifrontal/multifrontal method for asymmetric sparse matrices, Technical report TR-95-020, Computer and Information Science Department, University of Florida.

Davis T.A. and I.S. Duff (1993) An asymmetric-pattern multi-frontal method for sparse LU factorization.. Technical report RAL 93-036, Rutherford Appleton Laboratory.

Dennis Jr.,J.E and R.B. Schnabel 1983. Numerical methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall (Englewood Cliffs, New Jersey).

Dongarra J.J. 1995. Templates for the Solution of Systems of linear equations. Internet.

Dongarra, J.J., J. Du Croz, J.,I.S. Duff, and S. Hammarling 1990. A set of Level 3 Basic Linear Algebra Subproagrams, ACM Trans. Math.Softw. 16, 1-17.

DFDB 1994. Data Fitting Database. ASHRAE, Atlanta, GA

Duff I.S. and J.K. Reid 1995. MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems, Technical Report RAL 95-001, Rutherford Appleton Laboratory.

Duff Iain S. (1996) Sparse numerical linear algebra: direct methods and preconditioning'. Report TR-PA-96-22, CERFACS, 42 Ave G Cotriolis, 31057 Toulouse Cedex, France.

Dean R.H., and J.Ratzenberger 1985. Computer Simulation of VAV system Operation, HPAC, September

Dean R.H., F.J.Dean, and J.Ratzenberger 1985. Importance of Duct Design for VAV Systems, HPAC, August

Demuren, A.O. and F.G. Ideriah 1986. "Pipe network analysis by partial pivoting method." Journal of Hydraulic Engineering, Vol. 112, pp. 327-333.

Durfee, R.L. 1972. Measurement and Analysis of Leakage Rates from Seams and Joints of Air Handling Systems. AISI Project No. 1201-351/SMACNA Project No. 5-71

Evans, R.A. and R.J. Tsal 1996. Basic Tips for Duct Design, ASHRAE Journal, July, 1996

Eisenstat, S.C. and J.W.Liu 1992. Exploiting structural symmetry in asymmetric sparse symbolic factorization, SIAM J. Matrix Analysis and Applications 13, 203-211.

EPA 1995. Electric Sales and Revenue 1994, Energy Information Administration, Washington, DC

EPA 1985. Utility Cost Forecasting, Energy Information Administration, Washington

Furi M., Vignolli A. 1994. On the regularization on nonlinear ill-possed problem in Banach spaces. - J. Optimization Theory and Application, 4.3.

Hansen, P.C., T.Ostromsky, and Z.Zlatev 1994. Two enhancements in a partitioned sparse solver, in J.J.Dongarra and J.Wasniewski, eds, Parallel Scientific Computing. Proceedings of the PARA94 Conference, Copenhagen 1994, Lecture Notes in Mathematics 879, Springer, Berlin, pp. 296-303.

Hohmann A. Inexact Gauss Newton Methods for Parameter Dependent Nonlinear Problems. Technical report TR-93-13 (December 1993).

Markowitz, H.M. 1957. The elimination form of the inverse and its applications to linear programming, Management Science 3, 255-269.

Farajian, T., G. Grewal, R.J.Tsal 1992. Post-accident Air Leakage Analysis in a Nuclear Facility via T-Method Air flow Simulation. 22nd DOE/NDC Nuclear Air Cleaning and Treatment Conference, Denver

Farguhar, H.E. 1973. System Effect Values for Fans. ASHRAE Symposium Bulletin, Louisville, Kentucky

Fox, R.L. 1971. Optimization Methods for Engineering Design, Addison-Wesley Publishing Company, Reading, MA

George A. 1973. Nested dissection of a regular finite element mesh, SIAM J. Numerical Analysis 10, 345-363.

George, A. and L.W.H. Liu 1989. The evolution of the minimum degree ordering algorithm, SIAM Review 31(1), 1-19.

Gilbert, J.R. and T.Peierls 1988. Sparse partial pivoting in time proportional to arithmetic operations, SIAM J. Scientific and Statistical Computing 9, 862-874.

Graham, J.B. 1996. Duct System Pressure Gradient Diagrams and the Beer Cooler Problem, HPAC, August

Gregory, W.S., K.H. Duerre, and G.A. Bennett 1975. Ventilation system analysis during tornado conditions. Los Alamos Scientific Laboratory, Progress Report LA-6293-PR, July-December.

Hitchings, D.T. 1994. Laboratory Space Pressurization Control Systems, ASHRAE Journal, February

Horowitz, E. and S. Sahni 1976. Fundamentals of Data Structures. New York: Computer Science Press.

Houghton D.J., R. C. Bishop, A. B. Lovins, and B. L. Stickney, 1992. Space Cooling and Air Handling, E-Source Inc.report, Boulder, Colorado, August

Idelchik, I.E. 1996. Handbook of Hydraulic Resistance, 3rd edition, Begell Publishing House, Inc.

Isaacs, L.T., and K.G. Mills 1980. "Linear theory methods for pipe network analysis." Journal of the Hydraulics Division, Proceedings, ASCE, July, pp. 1191-1201

Jeppson, R.W. 1976. Analysis of flow in pipe network. Ann Arbor, MI: Ann Arbor Science.

Kamenev, P.N. 1938. Flow dynamics of industrial ventilation. Moscow: Stroisdat Publishing House.

Klote, J.H. 1981. " A Computer Program for Analysis of Smoke Control Systems." National Bureau of Standards, NBSIR 80-2157, Maryland

Konstantinov, U.M. 1981. Hydraulics. Kiev, USSR: Vischa Scola Publishing House.

Kovaric, M. 1971. Automatic Design of Optimal Duct Systems. Use of computers for environmental engineering related to buildings. Washington: National Bureau of Standards, Building Science Series 39, (October), pp.385-391.

Lam, C.F. and M.L.Walla, 1972. "Computer Analysis of Water Distribution Systems: Part I - Formulation of Equations", ASCE, Vol. 98.

Larson, E.D. and L.J. Nilsson 1991. Electricity Use in Pumping and Air Handling Systems, ASHRAE Transactions 97(2)

Leah, R.L., Pedersen, C.O, and Liebman, J.S. 1987. Optimization Using Quadratic Search - A Case Study of a Chilled Water System, ASHRAE Transactions, Vol.93, Part 2.

Liddament, M., and Thompson, C. 1982. "Mathematical models of air infiltration - a brief review and bibliography." Technical Note AIC 9, The Air Infiltration Centre, Bracknell, England.

Lobaev, B.N. 1959. Duct Calculation. Gosstroy Publishing House, Kiev, USSR. Available from the Library of Congress, Washington D.C.

Martin D.W. and G.Peters 1963. "The Application of Newton's Method to Network Analysis by Digital Computer" , Journal of Institute of Water Engineers, Vol.17

McDiarmid, Michael D. 1990. "Variable-Volume Fume Hoods: User Experience in a Chemistry Research Laboratory." ASHRAE Transactions, Vol. 96, Part 2.

Merenkov,A.P, 1992. “Mathematical modeling and optimization of heating/water/oil and gas supply systems,” Science Publishing House, Siberia branch, Novosibirsk, Russia.

Meyer, M.L. 1973. A New Concept: The Fan System Effect Factor, ASHRAE Symposium Bulletin, Louisville, KY

Moody, L.F. 1944. Friction factor for pipe flow. ASME Transactions, Vol.66, p.671.

Murtagh, B.A. 1972. An Approach to the Design of Networks. Chemical Engineering Science, Permagon Press, Vol.27, pp.1131-1143

Murphy G.C. 1976. Practical Aspects of Ductwork Design, ASHRAE Journal, October

Neuman, V.A., and Guven, H.M. 1988. "Laboratory Building HVAC Systems Optimization." ASHRAE Transactions 1988, Vol. 94, Part 2.

Neuman, V.A. and W.H. Rousseau 1986. " VAV for Laboratory Hoods - Design and Cost." ASHRAE Transections, Vol. 92, Part 1A.

Nocedal J. and S.Wright 1998. Numerical Optimization. Internet

Novak U., K.Weimann 1991. A Family of Newton Codes for Systems of Highly Nonlinear Equations. Technical Report TR-91-10 (December 1991).

Nillson, L.J. 1993. Energy Systems in Transition, Department of Environmental and Energy Systems Studies, Lund, Sweden

Numerical methods for Nonlinear Algebraic Equations. Edited by P.Rabinowitz. Gordon and Breach science publishers 1970.

Ortega J.M. and W.C.Rheinboldt 1970. Iterative solution of nonlinear equations in several variables. New York: Academic Press.

Ostrowski, A. 1966. Solution of equations and systems of equations, 2nd ed. New York: Academic Press. Peters G., and J.H. Wilkinson. The least squares problem and pseudo-inverses. Computer J., 13:309-316,1970.

Petzold L.R., K.E. Brenan, S.L. Campbell. 1989. The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations, Elsevier Science Publishing Co.

Rabinowitcz 1970. “Numerical methods for Nonlinear Algebraic Equations, Gordon and Breach science publishers.

Reid, J.K. 1987. Sparse matrices, in A. Iserles and M.J.D. Powell, eds, The State of the Art in Numerical Analysis, Oxford University Press, Oxford, pp. 59-85.

Rheinboldt W.C. 1982. On the computation of critical boundaries on equilibrium surfaces, SIAM J. Numer. Anal., 18 (1982), pp.653-669.

Rheinboldt W.C. and J.V. Burkardt 1983. Algorithm 596: A program for a locally parameterized continuation process, ACM Trans. Math. Software, No.9, pp. 236-241.

Rozell, J.M. 1974. Duct Turning Vanes in 90° Elbows, ASHRAE Transactions 80(2):53

Said, M.N.A. 1988. "A review of smoke control models." ASHRAE Journal, Vol. 30, No 4, April.

Scott K.B., 1986. Don't Ignore Duct Design for Optimized HVAC Systems, Specifying Engineer, No. 62, January

Shifrinson, B.L. 1937. New Method for District Water System Optimization, Heat and Power, No.2, Moscow, (Feb.), pp. 4-9. Available from the Library of Congress, Washington D.C.

SMACNA 1995. HVAC Duct Construction Standards - Metal and Flexible, 2nd Edition, Sheet Metal and Air Conditioning Contractor's Association, Chantilly, VA

SMACNA 1990. HVAC Systems - Duct Design, third edition

SMACNA 1985. HVAC Air Duct Leakage Test Manual

SMACNA 1975. Industry Practice for Industrial Duct Construction

SNIP 1976. II-33-75, Gosstroy, Stroysdat, Part II, Chapter 33, Moscow, USSR

Stoecker, W.F., R.C. Winn, and C.O. Pedersen 1971. Optimization of an Air-supply Duct System. Use of computers for environmental engineering related to buildings. National Bureau of Standards, Building Science Series 39, October

Stoecker, W. F., L. P. Rinck, and J. P. Wegrzyn 1974. Simulating flow rates, pressures, and temperatures in central chilled water systems. Second Symposium of the Use of Computers for Environmental Engineering Related to Buildings, Paris, June 13-15.

Swami, M.V., and Chandra, S. 1988. "Correlations for pressure distribution on buildings and calculation of natural-ventilation air flow." ASHRAE Transactions, Vol. 94, Part 1.

Swim, W.B. 1984. Analysis of Duct Leakage ETL Data from ASHRAE RP 308

Schnabel, R.B. and P Frank 1984. “Tensor methods for nonlinear equations”, SIAM J. Numer. Anal., 21, pp.815-843

Tihonov A.N., V.Y.Arsenin 1986. "The methods for ill-posed problems solution" Moscow, "Nauka"

T-DUCT Computer Program Manual 1994. Netsal & Associates, California

Tsal R.J., J. Bell 1998. "Laboratory Air Distribution." Laboratory Air Distribution Design Guide, CIEE and Lawrence Berkeley National Laboratory, San-Francisco, 1998

Tsal, R.J., H.F. Behls, and R. Mangel 1988. T-Method Duct Design, Part I: Optimization Theory, ASHRAE Transactions, Vol. 92, Part 1A

Tsal, R.J., H.F. Behls, and R. Mangel 1988. T-Method Duct Design, Part II: Calculation Procedure and Economic Analysis, ASHRAE Transactions, Vol. 92, Part 1A

Tsal, R.J., H.F. Behls, R. Mangel 1990. Duct Design Part, III: T-Method Duct Simulation. ASHRAE Transactions, Vol. 96, Part 2

Tsal, R.J., H.F. Behls, L.P.Varvak, 1998. T-Method Duct design. Part IV: Duct Leakage Theory, ASHRAE Transactions, Vol. 104, Part 2.

Tsal, R.J., H.F. Behls, L.P.Varvak 1998. T-Method Duct Design, Part V: Duct Leakage Calculation Technique and Economics. ASHRAE Transactions, Vol. 104, Part 2.

Tsal, R.J. and L.P. Varvak 1992. Duct Design Using the T-Method with Duct Leakage Incorporated. ASHRAE Research Project 641-RP

Tsal, R.J. and H.F. Behls 1990. Using the T-Method for Duct System Design, ASHRAE Journal, March

Tsal, R.J. and H.F. Behls 1989. "Ducting" In Technology Menu for Efficient End Use of Energy, University of Lund, Gerbagatan Lund, Sweden

Tsal, R.J. and P.W. Othmer 1989. New Method for Duct Design, CLIMA 2000. The Second International Congress on Heating, Refrigerating and Air Conditioning, Saraevo, Yugoslavia

Tsal, R.J. 1988. Calculation Techniques for Optimum Duct Design and Flow Simulation, ASHRAE Research Project 516-RP and Fluor Research Project 321563, Irvine, CA

Tsal, R.J., and H.F. Behls 1988. Fallacy of the Static Regain Duct Design Method, ASHRAE Technical Data Bulletin, ASHRAE, Atlanta, GA

Tsal, R.J. and M.S. Adler 1987. Evaluation of Numerical Methods for Ductwork and Pipeline Optimization, ASHRAE Transactions, Vol.93, Part 1

Tsal, R.J. and H.F. Behls 1986. Evaluation of Duct Design Methods. ASHRAE Transactions, Vol. 92, Part 1A

Tsal, R.J. and E.I. Chechik 1968. Use of Computers in HVAC Systems. Kiev: Budivelnick Publishing House. Available from the Library of Congress, Washington D.C. ,service number TD153.T77

Tsal, R.J. and N.Z. Shor 1967. The use of the steepest descent method for calculating flow distribution in a duct system. Economical cybernetics and operation research, No.2. Institute of Cybernetics Publication, Kiev, USSR

Tsao Tsu-Chin 1992. "Variable Air Volume Heating, Ventilation, and Air Condition (HVAC) Control System Model", Department of Mechanical and Industrial Engineering, University of Illinois, Urbana-Champaign, Technical report UILU-ENG 92-4032, Illinois

Vicente L.N. 1996. Trust-Region Interior Point Algorithms for a Class of Nonlinear Programming Problems, Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston, Texas

Walton, G.N. 1982. "Air flow and multi-room thermal analysis" ASHRAE Transactions, Vol. 88, Part 2.

Walton, G.N. 1984. "A computer algorithm for predicting infiltration and interroom air flows." ASHRAE Transactions, Vol. 90, Part 1.

Walton, G.N. 1989. " AIRNET - a Computer Program for Building Air flow Network Modeling", NISTIR 89-4072, National Institute of Standard and Technology, Maryland

Watson L. T. 1986. Numerical linear algebra aspects of globally convergent homotopy methods. SIAM Review Vol.28,No.4,December

Watson L. T. , S.C. Billups, and A.P. Morgan 1985. HOMPACK: A suite of codes for globally convergent homotopy algorithms, Tech. Rep. 85-34, Dept. of Industrial and Operations Eng., Univ. of Michigan, Ann Arbor, MI, 1985.

Wendes 1986. Sheet Metal Estimating, Wendes Engineering and Constructing Services, Inc.

Williams, G.J. 1995. Air Systems Basics, Heating/Piping/Air Conditioning, May

Wood, D.J., and A.G.Rayes 1981. "Reliability of algorithms for pipe network analysis." Journal of the Hydraulics Division, Vol. 107, pp. 1145-1161.